            IP Flow Information Export (IPFIX) Information Elements Extension
                          for TCP Connection Tracking
                     draft-fu-dots-ipfix-tcp-tracking-00

Abstract

   This document proposes several new TCP connection related Information
   Elements (IEs) for the IP Flow Information Export (IPFIX) protocol.
   The new Information Elements can be used to export certain
   characteristics regarding a TCP connection. Through massive
   gathering of such characteristics, it can help build an image of the
   TCP traffics passing through a network. The image will facilitate
   the detection of anomaly TCP traffic, especially attacks targeting at
   TCP.

Status of this Memo

Copyright and License Notice

Table of Contents

1. Introduction

   Due to its complex stateful operations, TCP [RFC0793] is especially
   vulnerable to attacks. The SYN Flood attack is an example, it
   involves with massive malicious clients attempting to set up
   connections with a server, but never completing the three-way
   handshake process, leaving the server-side of the connections in
   waiting states, eventually exhausting the server resources and no new
   connection can be created.

   Attack aiming at TCP can be low and slow in traffic pattern.
   Sometimes it may not take down the server, but just impair the
   provided service. Even though a victim server is still operating, its
   performance can be significantly degraded. Without the insight of
   what is going on with the TCP traffics, this kind of situation can be
   very hard to detect and analyze.

   For a network device, such as a router, to detect anomaly TCP
   traffics, it has to understand the semantics of TCP operations, more
   specifically, it has to be able to track TCP connection states. If a
   router has implemented such ability, it can export characteristics
   information regarding the TCP connections. By this way, offline
   analysis can be performed over the gathered information, which will
   facilitate the detection of anomaly TCP traffics, such as attacks.

   The IP Flow Information Export (IPFIX) protocol [RFC7011], already
   defines a generic mechanism for flow information export. This
   document introduces several new Information Elements of IPFIX, that
   can be used to export TCP connection characteristics. The proposed
   Information Elements are listed in Figure 1 below.

```
            +--------------------------+------+
            | Field Name               | IANA |
            |                          | IPFIX|
            |                          | ID   |
            +--------------------------+------+
            |tcpHandshakeSyn2SynAckTime | TBD  |
            |tcpHandshakeSynAck2AckTime | TBD  |
            |tcpHandshakeSyn2AckRttTime | TBD  |
            |tcpConnectionTrackingBits  | TBD  |
            |tcpPacketIntervalAverage   | TBD  |
            |tcpPacketIntervalVariance  | TBD  |
            |tcpOutOfOrderDeltaCount    | TBD  |
            +--------------------------+------+
```

              Figure 1: Information Element Table

   The Information Elements defined in Figure 1 are supposed to be

incorporated into the IANA IPFIX Information Elements registry
[IPFIX-IANA]. Their definitions can be found at Section 6.

2. 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 [RFC2119]

2.1. Terminology

IPFIX-specific terminology (Information Element, Template, Template
Record, Options Template Record, Template Set, Collector, Exporter,
Data Record, etc.) used in this document is defined in Section 2 of
[RFC7011]. As in [RFC7011], these IPFIX-specific terms have the
first letter of a word capitalized.

This document also makes use of the same terminology and definitions
as Section 2 of [RFC5470].

o Victim

The target that suffers from DDoS attack.

3. Connection Sampling and new IEs

3.1. Use Cases for New IEs

In this section, several use cases are discussed to identify the
requirements where new IEs are desirable for the network attacks
detection.

3.1.1. Response Time Calculation

For other DDoS attacks such as Http slowloris, there will be too many
connections that should be kept in the victim (server), which lead to
excessive resource consumption. As a result, the response time
between client and server will increase greatly. Challenge
Collapasar(CC) attack can also exhaust the resources of the server
and generate the similar results. Thus, the following IEs are
proposed as a symptom of these kinds of attacks:

tcpHandshakeSyn2SynAckTime: it denotes the time difference between
the time point that the Metering Process detects the SYN packet
from client to server and the time point that the observer views
the SYN-ACK packet from server to client.

tcpHandshakeSynAck2AckTime: it denotes the time difference between the time point that the Metering Process detects the SYN-ACK packet from server to client and the time point that the observer views the ACK packet from client to server.

tcpHandshakeSyn2AckRttTime: it denotes The sum of tcpHandshakeSyn2SynAckTime and tcpHandshakeSynAck2AckTime. It is the Round Trip Time (RTT) between client and server.

3.1.2. Symptoms of Exceptions

In http slowloris attack the client may send packets to victim periodically which can cause the performance lost on the server. The characteristic of the attack is that there are too many connections on the victim. However, the traffic volume for these connections is small. In order to detect this attack, the first step is to get the packets that are belonging to the same connection. The second step is to find the periodicity. Thus the two indices tcpPacketIntervalAverage and tcpPacketIntervalVariance are needed. The index tcpPacketIntervalAverage denotes the average time difference between two successive packets and the index tcpPacketIntervalVariance denotes the variance of multiple time difference. Large tcpPacketIntervalAverage and small tcpPacketIntervalVariance can be a symptom of slow packet attack, since the attacker sends packets in large intervals just as to keep the connection open, and the intervals tend to differ very little in time.

To degrade the performance of the victim, the malicious clients may send too many out-of-order packets, which will consume too much memory on the server. Although out-of-order packets are permit in the TCP protocol, it is possible to be leveraged to cause DDoS attack. So the index tcpOutOfOrderDeltaCount is helpful to detect this kind of exception. For observer, it maintains one counter for each TCP connection. The initial sequence number of the client is saved in the counter. The counter increases by the sequence number of the packets it sees from client to server. If the observer sees a packet with lower sequence number than the current counter value, then the packet will be considered as an out-of-order packet.

In IPFIX, the index tcpControlBits is used to record the corresponding status bits in TCP header of the packets[IPFIX-IANA]. In order to detect the application attacks which can cause the protocol exception such as the wrong use of the TCP status bits before and after the TCP connection establishment, another index called tcpConnectionTrackingBits is needed. For example, when the observer sees the SYN packet from client to server, it sets 15th bit of tcpConnectionTrackingBits to 1; when it sees the SYN-ACK packet

from server to client, it sets 14th bit to 1, and so on. If one
endpoint sends the packet with wrong bits during the establishment of
the connection, then the observer will identify the exception by the
value of tcpConnectionTrackingBits.

4. Application of the New IEs for Attack Detection

    This section presents a number of examples to help for the easy
    understanding of the application of these new IEs for attack
    detection.

4.1. Detect Slowloris Attack

    The template for detecting resource exhausting application attack
    such as http slowloris attack should contain a subnet of IEs shown in
    Table 4.

```
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |           Set ID = 2          |         Length = 48 octets    |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |         Template ID TBD       |          Field Count = 10     |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0|     sourceIPv4Address       |         Field Length = 4      |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0| destinationIPv4Address      |         Field Length = 4      |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0| protocolIdentifier          |         Field Length = 1      |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0| tcpHandshakeSyn2SynAckTime   |         Field Length = 2      |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0| tcpHandshakeSynAck2AckTime   |         Field Length = 2      |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0| tcpHandshakeSyn2SynAckTime   |         Field Length = 2      |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0| tcpPacketIntervalAverage    |         Field Length = 4      |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0| tcpPacketIntervalVariance   |         Field Length = 4      |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0|     flowStartSeconds        |         Field Length = 4      |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0|     flowEndSeconds          |         Field Length = 4      | +-
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

              Figure 2: Template example for detecting slowloris attack

    An example of the actual record is shown below in a readable form as
    below:

    {sourceIPv4Address = 192.168.0.101, destinationIPv4Address =
    192.168.0.201, protocolIdentifier = 6, tcpHandshakeSyn2SynAckTime =
    200, tcpHandshakeSynAck2AckTime = 10, tcpHandshakeSyn2AckRttTime =
    210, tcpPacketIntervalAverage = 500, tcpPacketIntervalVariance =
    1000, flowStartSeconds = 100, flowEndSeconds = 200}

4.2. Detect Out-of-order Packets Attack

    The template for detecting out-of-order packets attack should contain
    IEs shown in Table 5.

```
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |           Set ID = 2          |       Length = 32 octets      |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |         Template ID TBD       |        Field Count = 10       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0|      sourceIPv4Address      |        Field Length = 4       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0| destinationIPv4Address      |        Field Length = 4       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0|      protocolIdentifier     |        Field Length = 1       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0|      packetDeltaCount       |        Field Length = 8       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0|   tcpOutOfOrderDeltaCount   |        Field Length = 4       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0|      flowStartSeconds       |        Field Length = 4       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |0|      flowEndSeconds         |        Field Length = 4       |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

        Figure 3: Template example for detecting out-of-order attack

    An example of the actual record is shown below in a readable form as
    below:

    {sourceIPv4Address = 192.168.0.101, destinationIPv4Address =
    192.168.0.201, protocolIdentifier = 6, packetDeltaCount =3000,
    tcpOutOfOrderDeltaCount = 2000, flowStartSeconds = 100,
    flowEndSeconds = 200}

5. Security Considerations

    No additional security considerations are introduced in this
    document. The same security considerations as for the IPFIX protocol
    [RFC7011] apply.

6. IANA Considerations

The following information elements are requested from IANA IPFIX
registry. Upon acceptance, the 'TBD' values of the ElementIds
should be replaced by IANA for assigned numbers.

Name: tcpHandshakeSyn2SynAckTime
Description:
    The time difference between a SYN and its corresponding SYN-ACK
    when the Metering Process observes a new TCP connection is
    going to be set up.
Abstract Data Type: dateTimeMicroseconds
ElementId: TBD
Status: current
Units: microseconds

Name: tcpHandshakeSynAck2AckTime
Description:
    The time difference between a SYN-ACK and its corresponding ACK
    when the Metering Process observes a new TCP connection is
    going to be set up.
Abstract Data Type: dateTimeMicroseconds
ElementId: TBD
Status: current
Units: microseconds

Name: tcpHandshakeSyn2AckRttTime
Description:
    The time difference between a SYN and its corresponding ACK
    sent from the same endpoint when the Metering Process observes
    a new TCP connection is going to be set up.

    Conceptually tcpHandshakeSyn2AckRttTime can be thought as the
    sum of tcpHandshakeSyn2SynAckTime and
    tcpHandshakeSynAck2AckTime, but practically the values may
    differ.
Abstract Data Type: dateTimeMicroseconds
ElementId: TBD
Status: current
Units: microseconds

Name: tcpConnectionTrackingBits
Description:
    These bits are used by the Metering Process to track a TCP
    connection. A bit is set to 1 if the corresponding condition
    is met. A value of 0 for a bit indicates the corresponding
    condition was net met.

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1|1|1|1|1|1|0|0|0|0|0|0|0|0|0|0|
|5|4|3|2|1|0|9|8|7|6|5|4|3|2|1|0|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|S|S|A|F|A|F|A|R|T|E|END|R|R|E|V|
|Y|/|C|I|C|/|C|S|M|N|REA|O|O|R|L|
|N|A|K|N|K|A|K|T|R|D|SON|P|D|R|D|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Bit 15 (SYN):
   Set when there is no TCP connection between the endpoints
   and the Metering Process detects a SYN as it is used to
   setup a new TCP connection. The Metering Process starts to
   track the TCP connection.
Bit 14 (S/A):
   Set when bit 15 has been set and the Metering Process
   detects a SYN-ACK in the flow, which effectively
   acknowledges the SYN causing bit 15 to be set.
Bit 13 (ACK):
   Set when bit 15 and bit 14 have been set and the Metering
   Process detects an ACK which effectively acknowledges the
   SYN causing bit 14 to be set. Upon setting this bit, it
   means handshake of the TCP connection setup has completed.
Bit 12 (FIN):
   Set when the Metering Process detects the first FIN for the
   established and tracked TCP connection. It means the TCP
   connection is going to be closed.
Bit 11 (ACK):
   Set when bit 12 has been set and the Metering Process
   detects an ACK which effectively acknowledges the FIN
   causing bit 12 to be set.
Bit 10 (F/A):
   Set when bit 12 has been set and the Metering Process
   detects a FIN that is from the opposite of the endpoint
   which sent the FIN causing bit 12 to be set.
Bit 09 (ACK):
   Set when bit 10 has been set and the Metering Process
   detects an ACK that is from the same endpoint which sent the
   FIN causing bit 10 to be set.
Bit 08 (RST):
   Set when the Metering Process detects any RST from either
   party of the tracked TCP connection.
Bit 07 (TMR):
   Set when a flow record report is triggered by a periodic
   reporting timer. It means the TCP connection is still under
   tracking.
Bit 06 (END):
   Set when the Metering Process has stopped tracking the TCP

          connection, as the connection has been closed or aborted.
     Bit 05 & Bit 04 (END REASON):
        00: as default value or the tracked TCP connection
            is closed.
        01: the tracked TCP connection is aborted.
        10: the tracked TCP connection is inactive after a period
            of time.
        11: reserved.
     Bit 03 (ROP):
        Set when the Metering Process detects any SYN or SYNACK,
        after the both endpoints have sent FIN or an RST has been
        detected.
     Bit 02 (ROD):
        Set when the Metering Process detects at least 50 TCP
        segments being exchanged, after both endpoints have sent FIN
        or an RST has been detected.
     Bit 01 (ERR):
        Set when the Metering Process detects any of the following
        abnormal signaling sequences for the TCP connection:
        SYN/FIN, SYN/FIN/PSH, SYN/FIN/RST, SYN/FIN/RST/PSH.
     Bit 00 (VLD):
        Set when the tracked TCP connection is closed normally.

  Abstract Data Type: unsigned16
  Data Type Semantics: flags
  ElementId: TBD
  Status: current

  Name: tcpPacketIntervalAverage
  Description:
     The average time interval calculated by the Metering Process
     between two successive packets in the data flow of a TCP
     connection.
  Abstract Data Type: unsigned32
  ElementId: TBD
  Status: current

  Name: tcpPacketIntervalVariance
  Description:
     The variance of the time intervals calculated by the Metering
     Process between two successive packets in the data flow of a
     TCP connection.
  Abstract Data Type: unsigned64
  ElementId: TBD
  Status: current

          Name: tcpOutOfOrderDeltaCount
          Description:
             The number of out of order packets in the data flow of a TCP
             connection detected at the Observation Point since the previous
             report.
          Abstract Data Type: unsigned64
          Data Type Semantics: deltaCounter
          ElementId: TBD
          Status: current

7. References

7.1. Normative References

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

    [RFC7011] Claise, B., Trammell, B., and P. Aitken, "Specification
               of the IP Flow Information Export (IPFIX) Protocol for the
               Exchange of Flow Information", STD 77, RFC 7011, September
               2013.

    [RFC5470] Sadasivan, G., Brownlee, N., Claise, B., and J. Quittek,
               "Architecture for IP Flow Information Export", RFC 5470,
               March 2009.

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

7.2. Informative References

    [IPFIX-IANA]
               IANA, "IPFIX Information Elements registry",
                 <http://www.iana.org/assignments/ipfix>.

    [RFC5474] Duffield, N., Ed., Chiou, D., Claise, B., Greenberg, A.,
               Grossglauser, M., and J. Rexford, "A Framework for Packet
               Selection and Reporting", RFC 5474, March 2009.

    [RFC5475] Zseby, T., Molina, M., Duffield, N., Niccolini, S., and F.
               Raspall, "Sampling and Filtering Techniques for IP Packet
               Selection", RFC 5475, March 2009.

    [RFC5476] Claise, B., Ed., Johnson, A., and J. Quittek, "Packet
               Sampling (PSAMP) Protocol Specifications", RFC 5476, March
               2009.

   [RFC5477] Dietz, T., Claise, B., Aitken, P., Dressler, F., and G.
              Carle, "Information Model for Packet Sampling Exports",
              RFC 5477, March 2009,

8. Acknowledgments

Authors' Addresses

   Tianfu Fu
   Huawei
   Q11, Huanbao Yuan, 156 Beiqing Road, Haidian District
   Beijing 100095
   China

   Email: futianfu@huawei.com

   Chong Zhou
   Huawei

   156 Beiqing Road, M06 Shichuang Technology Demonstration Park
   Haidian, Beijing 100094
   China

   Email: mr.zhouchong@huawei.com

   Hui Zheng (Marvin)
   Huawei

   101 Software Avenue, Yuhuatai District
   Nanjing, Jiangsu 210012
   China

   Email: marvin.zhenghui@huawei.com

DOTS   A. Mortensen
Internet-Draft   Arbor Networks, Inc.
Intended status: Informational   F. Andreasen
Expires: May 4, 2017   T. Reddy
                                                       Cisco Systems, Inc.
                                                                  C. Gray
                                                            Comcast, Inc.
                                                              R. Compton
                                          Charter Communications, Inc.
                                                               N. Teague
                                                           Verisign, Inc.
                                                         October 31, 2016

Distributed-Denial-of-Service Open Threat Signaling (DOTS) Architecture
                  draft-ietf-dots-architecture-01

Abstract

   This document describes an architecture for establishing and
   maintaining Distributed Denial of Service (DDoS) Open Threat
   Signaling (DOTS) within and between domains. The document does not
   specify protocols or protocol extensions, instead focusing on
   defining architectural relationships, components and concepts used in
   a DOTS deployment.

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 May 4, 2017.

Copyright Notice

Table of Contents

1. Context and Motivation

    Signaling the need for help defending against an active distributed
    denial of service (DDoS) attack requires a common understanding of
    mechanisms and roles among the parties coordinating defensive
    response. The signaling layer and supplementary messaging is the
    focus of DDoS Open Threat Signaling (DOTS). DOTS defines a method of
    coordinating defensive measures among willing peers to mitigate
    attacks quickly and efficiently, enabling hybrid attack responses
    coordinated locally at or near the target of an active attack, or
    anywhere in-path between attack sources and target.

    This document describes an architecture used in establishing,
    maintaining or terminating a DOTS relationship within a domain or
    between domains.

1.1. Terminology

1.1.1. Key Words

    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].

1.1.2. Definition of Terms

    This document uses the terms defined in [I-D.ietf-dots-requirements].

1.2. Scope

    In this architecture, DOTS clients and servers communicate using the
    DOTS signaling. As a result of signals from a DOTS client, the DOTS
    server may modify the forwarding path of traffic destined for the
    attack target(s), for example by diverting traffic to a mitigator or
    pool of mitigators, where policy may be applied to distinguish and
    discard attack traffic. Any such policy is deployment-specific.

    The DOTS architecture presented here is applicable across network
    administrative domains - for example, between an enterprise domain
    and the domain of a third-party attack mitigation service - as well
    as to a single administrative domain. DOTS is generally assumed to
    be most effective when aiding coordination of attack response between
    two or more participating network domains, but single domain
    scenarios are valuable in their own right, as when aggregating intra-
    domain DOTS client signals for inter-domain coordinated attack
    response.

This document does not address any administrative or business
agreements that may be established between involved DOTS parties.
Those considerations are out of scope. Regardless, this document
assumes necessary authentication and authorization mechanism are put
in place so that only authorized clients can invoke the DOTS service.

1.3. Assumptions

This document makes the following assumptions:

o All domains in which DOTS is deployed are assumed to offer the
   required connectivity between DOTS agents and any intermediary
   network elements, but the architecture imposes no additional
   limitations on the form of connectivity.

o Congestion and resource exhaustion are intended outcomes of a DDoS
   attack [RFC4732]. Some operators may utilize non-impacted paths
   or networks for DOTS, but in general conditions should be assumed
   to be hostile and that DOTS must be able to function in all
   circumstances, including when the signaling path is significantly
   impaired.

o There is no universal DDoS attack scale threshold triggering a
   coordinated response across administrative domains. A network
   domain administrator, or service or application owner may
   arbitrarily set attack scale threshold triggers, or manually send
   requests for mitigation.

o Mitigation requests may be sent to one or more upstream DOTS
   servers based on criteria determined by DOTS client
   administrators. The number of DOTS servers with which a given
   DOTS client has established signaling sessions is determined by
   local policy and is deployment-specific.

o The mitigation capacity and/or capability of domains receiving
   requests for coordinated attack response is opaque to the domains
   sending the request. The domain receiving the DOTS client signal
   may or may not have sufficient capacity or capability to filter
   any or all DDoS attack traffic directed at a target. In either
   case, the upstream DOTS server may redirect a request to another
   DOTS server. Redirection may be local to the redirecting DOTS
   server's domain, or may involve a third-party domain.

o DOTS client and server signals, as well as messages sent through
   the data channel, are sent across any transit networks with the
   same probability of delivery as any other traffic between the DOTS
   client domain and the DOTS server domain. Any encapsulation
   required for successful delivery is left untouched by transit

network elements. DOTS server and DOTS client cannot assume any
preferential treatment of DOTS signals. Such preferential
treatment may be available in some deployments, and the DOTS
architecture does not preclude its use when available. However,
DOTS itself does not address how that may be done.

o The architecture allows for, but does not assume, the presence of
   Quality of Service (QoS) policy agreements between DOTS-enabled
   peer networks or local QoS prioritization aimed at ensuring
   delivery of DOTS messages between DOTS agents. QoS is an
   operational consideration only, not a functional part of the DOTS
   architecture.

o The signal channel and the data channel may be loosely coupled,
   and need not terminate on the same DOTS server.

2. Architecture

   The basic high-level DOTS architecture is illustrated in Figure 1:

```
      +-----------+                +------------+
      | Mitigator | ~~~~~~~~~~ | DOTS Server |
      +-----------+                +------------+
                                          |
                                          |
                                          |
                                          |
      +---------------+            +------------+
      | Attack Target | ~~~~~ | DOTS Client |
      +---------------+            +------------+
```

                   Figure 1: Basic DOTS Architecture

   A simple example instantiation of the DOTS architecture could be an
   enterprise as the attack target for a volumetric DDoS attack, and an
   upstream DDoS mitigation service as the Mitigator. The enterprise
   (attack target) is connected to the Internet via a link that is
   getting saturated, and the enterprise suspects it is under DDoS
   attack. The enterprise has a DOTS client, which obtains information
   about the DDoS attack, and signals the DOTS server for help in
   mitigating the attack. The DOTS server in turn invokes one or more
   mitigators, which are tasked with mitigating the actual DDoS attack,
   and hence aim to suppress the attack traffic while allowing valid
   traffic to reach the attack target.

   The scope of the DOTS specifications is the interfaces between the
   DOTS client and DOTS server. The interfaces to the attack target and
   the mitigator are out of scope of DOTS. Similarly, the operation of
   both the attack target and the mitigator are out of scope of DOTS.

Thus, DOTS neither specifies how an attack target decides it is under
DDoS attack, nor does DOTS specify how a mitigator may actually
mitigate such an attack. A DOTS client's request for mitigation is
advisory in nature, and may not lead to any mitigation at all,
depending on the DOTS server domain's capacity and willingness to
mitigate on behalf of the DOTS client's domain.

As illustrated in Figure 2, there are two interfaces between the DOTS
server and the DOTS client:

```
  +---------------+                                +---------------+
  |               | <------- Signal Channel ------> |               |
  |  DOTS Client  |                                |  DOTS Server  |
  |               | <======  Data Channel  ======> |               |
  +---------------+                                +---------------+
```
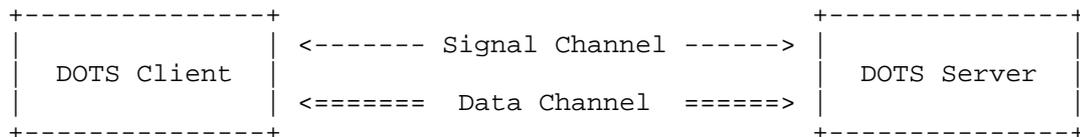
                        Figure 2: DOTS Interfaces

The DOTS client may be provided with a list of DOTS servers, each
associated with one or more IP addresses. These addresses may or may
not be of the same address family. The DOTS client establishes one
or more signaling sessions by connecting to the provided DOTS server
addresses.

[[EDITOR'S NOTE: We request feedback from the working group about the
mechanism of server discovery.]]

The primary purpose of the signal channel is for a DOTS client to ask
a DOTS server for help in mitigating an attack, and for the DOTS
server to inform the DOTS client about the status of such mitigation.
The DOTS client does this by sending a client signal, which contains
information about the attack target or targets. The client signal
may also include telemetry information about the attack, if the DOTS
client has such information available. The DOTS server in turn sends
a server signal to inform the DOTS client of whether it will honor
the mitigation request. Assuming it will, the DOTS server initiates
attack mitigation (by means outside of DOTS), and periodically
informs the DOTS client about the status of the mitigation.
Similarly, the DOTS client periodically informs the DOTS server about
the client's status, which at a minimum provides client (attack
target) health information, but it may also include telemetry
information about the attack as it is now seen by the client. At
some point, the DOTS client may decide to terminate the server-side
attack mitigation, which it indicates to the DOTS server over the
signal channel. A mitigation may also be terminated if a DOTS
client-specified mitigation time limit is exceeded; additional
considerations around mitigation time limits may be found below.
Note that the signal channel may need to operate over a link that is

experiencing a DDoS attack and hence is subject to severe packet loss
and high latency.

While DOTS is able to request mitigation with just the signal
channel, the addition of the DOTS data channel provides for
additional and more efficient capabilities; both channels are
required in the DOTS architecture. The primary purpose of the data
channel is to support DOTS related configuration and policy
information exchange between the DOTS client and the DOTS server.
Examples of such information include, but are not limited to:

o Creating identifiers, such as names or aliases, for resources for
   which mitigation may be requested. Such identifiers may then be
   used in subsequent signal channel exchanges to refer more
   efficiently to the resources under attack, as seen in Figure 3
   below, using JSON to serialize the data:

```
        {
            "https1": [
                "192.0.2.1:443",
                "198.51.100.2:443",
            ],
            "proxies": [
                "203.0.113.3:3128",
                "[2001:DB8:AC10::1]:3128"
            ],
            "api_urls": "https://apiserver.example.com/api/v1",
        }
```

                 Figure 3: Protected resource identifiers

o Black-list management, which enables a DOTS client to inform the
   DOTS server about sources to suppress.

o White-list management, which enables a DOTS client to inform the
   DOTS server about sources from which traffic should always be
   accepted.

o Filter management, which enables a DOTS client to install or
   remove traffic filters dropping or rate-limiting unwanted traffic.

o DOTS client provisioning.

Note that while it is possible to exchange the above information
before, during or after a DDoS attack, DOTS requires reliable
delivery of the this information and does not provide any special
means for ensuring timely delivery of it during an attack. In

practice, this means that DOTS deployments SHOULD NOT rely on such
information being exchanged during a DDoS attack.

2.1. DOTS Operations

The scope of DOTS is focused on the signaling and data exchange
between the DOTS client and DOTS server. DOTS does not prescribe any
specific deployment models, however DOTS is designed with some
specific requirements around the different DOTS agents and their
relationships.

First of all, a DOTS agent belongs to an domain, and that domain has
an identity which can be authenticated and authorized. DOTS agents
communicate with each other over a mutually authenticated signal
channel and data channel. However, before they can do so, a service
relationship needs to be established between them. The details and
means by which this is done is outside the scope of DOTS, however an
example would be for an enterprise A (DOTS client) to sign up for
DDoS service from provider B (DOTS server). This would establish a
(service) relationship between the two that enables enterprise A's
DOTS client to establish a signal channel with provider B's DOTS
server. A and B will authenticate each other, and B can verify that
A is authorized for its service.

From an operational and design point of view, DOTS assumes that the
above relationship is established prior to a request for DDoS attack
mitigation. In particular, it is assumed that bi-directional
communication is possible at this time between the DOTS client and
DOTS server. Furthermore, it is assumed that additional service
provisioning, configuration and information exchange can be performed
by use of the data channel, if operationally required. It is not
until this point that the mitigation service is available for use.

Once the mutually authenticated signal channel has been established,
it will remain in place. This is done to increase the likelihood
that the DOTS client can signal the DOTS server for help when the
attack target is being flooded, and similarly raise the probability
that DOTS server signals reach the client regardless of inbound link
congestion. This does not necessarily imply that the attack target
and the DOTS client have to be co-located in the same administrative
domain, but it is expected to be a common scenario.

DDoS mitigation service with the help of an upstream mitigator will
often involve some form of traffic redirection whereby traffic
destined for the attack target is diverted towards the mitigator,
e.g. by use of BGP [RFC4271] or DNS [RFC1034]. The mitigator in turn
inspects and scrubs the traffic, and forwards the resulting
(hopefully non-attack) traffic to the attack target. Thus, when a

DOTS server receives an attack mitigation request from a DOTS client,
it can be viewed as a way of causing traffic redirection for the
attack target indicated.

DOTS relies on mutual authentication and the pre-established service
relationship between the DOTS client's domain and the DOTS server's
domain to provide basic authorization. The DOTS server SHOULD
enforce additional authorization mechanisms to restrict the
mitigation scope a DOTS client can request, but such authorization
mechanisms are deployment-specific.

Although co-location of DOTS server and mitigator within the same
domain is expected to be a common deployment model, it is assumed
that operators may require alternative models. Nothing in this
document precludes such alternatives.

2.2. Components

2.2.1. DOTS Client

A DOTS client is a DOTS agent from which requests for help
coordinating attack response originate. The requests may be in
response to an active, ongoing attack against a target in the DOTS
client's domain, but no active attack is required for a DOTS client
to request help. Local operators may wish to have upstream
mitigators in the network path for an indefinite period, and are
restricted only by business relationships when it comes to duration
and scope of requested mitigation.

The DOTS client requests attack response coordination from a DOTS
server over the signal channel, including in the request the DOTS
client's desired mitigation scoping, as described in
[I-D.ietf-dots-requirements]. The actual mitigation scope and
countermeasures used in response to the attack are up to the DOTS
server and Mitigator operators, as the DOTS client may have a narrow
perspective on the ongoing attack. As such, the DOTS client's
request for mitigation should be considered advisory: guarantees of
DOTS server availability or mitigation capacity constitute service
level agreements and are out of scope for this document.

The DOTS client adjusts mitigation scope and provides available
attack details at the direction of its local operator. Such
direction may involve manual or automated adjustments in response to
feedback from the DOTS server.

To provide a metric of signal health and distinguish an idle
signaling session from a disconnected or defunct session, the DOTS
client sends a heartbeat over the signal channel to maintain its half

of the signaling session. The DOTS client similarly expects a
heartbeat from the DOTS server, and MAY consider a signaling session
terminated in the extended absence of a DOTS server heartbeat.

2.2.2. DOTS Server

   A DOTS server is a DOTS agent capable of receiving, processing and
   possibly acting on requests for help coordinating attack response
   from one or more DOTS clients. The DOTS server authenticates and
   authorizes DOTS clients as described in Signaling Sessions below, and
   maintains signaling session state, tracking requests for mitigation,
   reporting on the status of active mitigations, and terminating
   signaling sessions in the extended absence of a client heartbeat or
   when a session times out.

   Assuming the preconditions discussed below exist, a DOTS client
   maintaining an active signaling session with a DOTS server may
   reasonably expect some level of mitigation in response to a request
   for coordinated attack response.

   The DOTS server enforces authorization of DOTS clients' signals for
   mitigation. The mechanism of enforcement is not in scope for this
   document, but is expected to restrict requested mitigation scope to
   addresses, prefixes, and/or services owned by the DOTS client's
   administrative domain, such that a DOTS client from one domain is not
   able to influence the network path to another domain. A DOTS server
   MUST reject requests for mitigation of resources not owned by the
   requesting DOTS client's administrative domain. A DOTS server MAY
   also refuse a DOTS client's mitigation request for arbitrary reasons,
   within any limits imposed by business or service level agreements
   between client and server domains. If a DOTS server refuses a DOTS
   client's request for mitigation, the DOTS server SHOULD include the
   refusal reason in the server signal sent to the client.

   A DOTS server is in regular contact with one or more mitigators. If
   a DOTS server accepts a DOTS client's request for help, the DOTS
   server forwards a translated form of that request to the mitigator or
   mitigators responsible for scrubbing attack traffic. Note that the
   form of the translated request passed from the DOTS server to the
   mitigator is not in scope: it may be as simple as an alert to
   mitigator operators, or highly automated using vendor or open
   application programming interfaces supported by the mitigator. The
   DOTS server MUST report the actual scope of any mitigation enabled on
   behalf of a client.

   The DOTS server SHOULD retrieve available metrics for any mitigations
   activated on behalf of a DOTS client, and SHOULD include them in

   server signals sent to the DOTS client originating the request for
   mitigation.

   To provide a metric of signal health and distinguish an idle
   signaling session from a disconnected or defunct session, the DOTS
   server sends a heartbeat over the signal channel to maintain its half
   of the signaling session. The DOTS server similarly expects a
   heartbeat from the DOTS client, and MAY consider a signaling session
   terminated in the extended absence of a DOTS client heartbeat.

2.2.3. DOTS Gateway

   Traditional client to server relationships may be expanded by
   chaining DOTS sessions. This chaining is enabled through "logical
   concatenation" [RFC7092] of a DOTS server and a DOTS client,
   resulting in an application analogous to the SIP logical entity of a
   Back-to-Back User Agent (B2BUA) [RFC3261]. The term DOTS gateway
   will be used here and the following text will describe some
   interactions in relation to this application.

   A DOTS gateway may be deployed client-side, server-side or both. The
   gateway may terminate multiple discrete client connections and may
   aggregate these into a single or multiple DOTS signaling sessions.

   The DOTS gateway will appear as a server to its downstream agents and
   as a client to its upstream agents, a functional concatenation of the
   DOTS client and server roles, as depicted in Figure 4:

```
                         +-------------+
                         |      | D |      |
         +----+          |      | O |      |         +----+
         | c1 |----------| s1   | T | c2   |---------| s2 |
         +----+          |      | S |      |         +----+
                         |      | G |      |
                         +-------------+
```

                          Figure 4: DOTS gateway

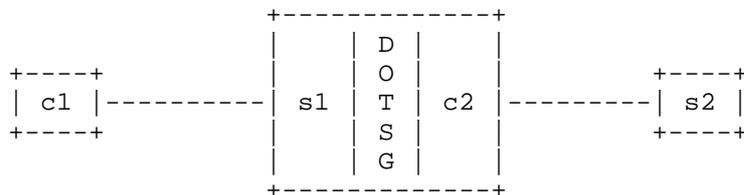   The DOTS gateway performs full stack DOTS session termination and
   reorigination between its client and server side. The details of how
   this is achieved are implementation specific. The DOTS protocol does
   not include any special features related to DOTS gateways, and hence
   from a DOTS perspective, whenever a DOTS gateway is present, the DOTS
   session simply terminates/originates there.

2.3. DOTS Agent Relationships

   So far, we have only considered a relatively simple scenario of a
   single DOTS client associated with a single DOTS server, however DOTS
   supports more advanced relationships.

   A DOTS server may be associated with one or more DOTS clients, and
   those DOTS clients may belong to different domains. An example
   scenario is a mitigation provider serving multiple attack targets
   (Figure 5):

```
      DOTS Clients        DOTS Server
      +---+
      | c |-----------
      +---+            \
      example.org       \
                         \
      +---+               \ +---+
      | c |----------------| S |
      +---+               / +---+
      example.com        /
                        /
      +---+            /
      | c |-----------
      +---+
      example.com         example.net
```

                  Figure 5: DOTS server with multiple clients

   A DOTS client may be associated with one or more DOTS servers, and
   those DOTS servers may belong to different domains. This may be to
   ensure high availability or co-ordinate mitigation with more than one
   directly connected ISP. An example scenario is for an enterprise to
   have DDoS mitigation service from multiple providers, as shown in
   Figure 6 below. Operational considerations relating to co-ordinating
   multiple provider responses are beyond the scope of DOTS.

   [[EDITOR'S NOTE: we request working group feedback and discussion of
   operational considerations relating to coordinating multiple provider
   responses to a mitigation request.]]

```
        DOTS Client        DOTS Servers
                            +---+
             -----------| S |
            /               +---+
           /                dots1.example.net
          /
    +---+/              +---+
    | c |---------------| S |
    +---+\              +---+
         \              dots.example.org
          \
           \            +---+
            -----------| S |
                        +---+
        example.com     dots2.example.net
```
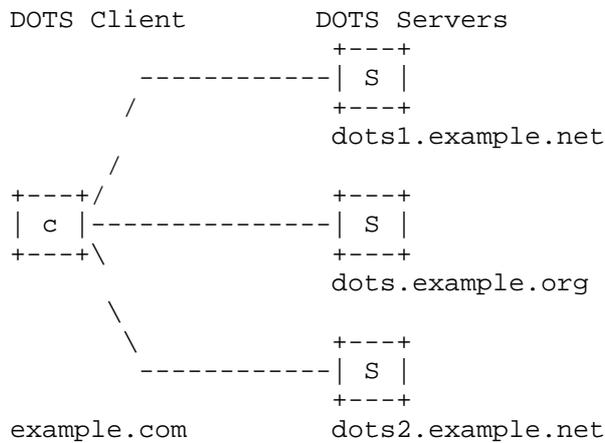
                    Figure 6: Multi-Homed DOTS Client

2.3.1. Gatewayed signaling

   As discussed above in Section 2.2.3, a DOTS gateway is a logical
   function chaining signaling sessions through concatenation of a DOTS
   server and DOTS client.

   An example scenario, as shown in Figure 7 and Figure 8 below, is for
   an enterprise to have deployed multiple DOTS capable devices which
   are able to signal intra-domain using TCP [RFC0793] on un-congested
   links to a DOTS gateway which may then transform these to a UDP
   [RFC0768] transport inter-domain where connection oriented transports
   may degrade; this applies to the signal channel only, as the data
   channel requires a connection-oriented transport. The relationship
   between the gateway and its upstream agents is opaque to the initial
   clients.

```
     +---+
     | c |\
     +---+ \               +---+
            \-----TCP-----| D |                +---+
     +---+                | O |                |   |
     | c |--------TCP-----| T |------UDP------| S |
     +---+                | S |                |   |
            /-----TCP-----| G |                +---+
     +---+ /               +---+
     | c |/
     +---+
     example.com          example.com          example.net
     DOTS Clients         DOTS Gateway (DOTSG)  DOTS Server
```
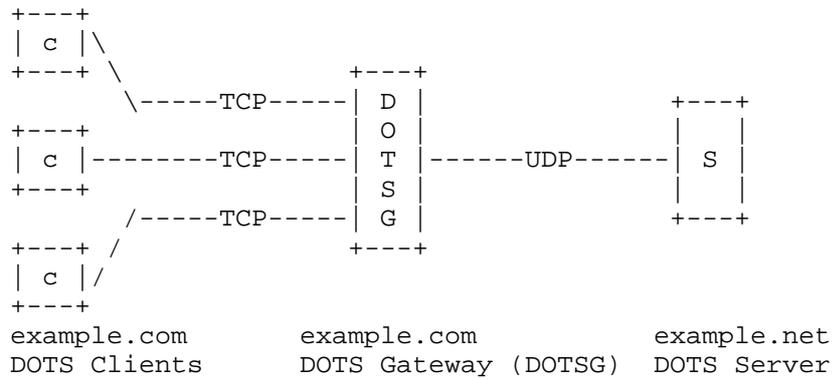
                Figure 7: Client-Side Gateway with Aggregation

```
     +---+
     | c |\
     +---+ \               +---+
            \-----TCP-----| D |------UDP------+---+
     +---+                | O |                |   |
     | c |--------TCP-----| T |------UDP------| S |
     +---+                | S |                |   |
            /-----TCP-----| G |------UDP------+---+
     +---+ /               +---+
     | c |/
     +---+
     example.com          example.com          example.net
     DOTS Clients         DOTS Gateway (DOTSG)  DOTS Server
```
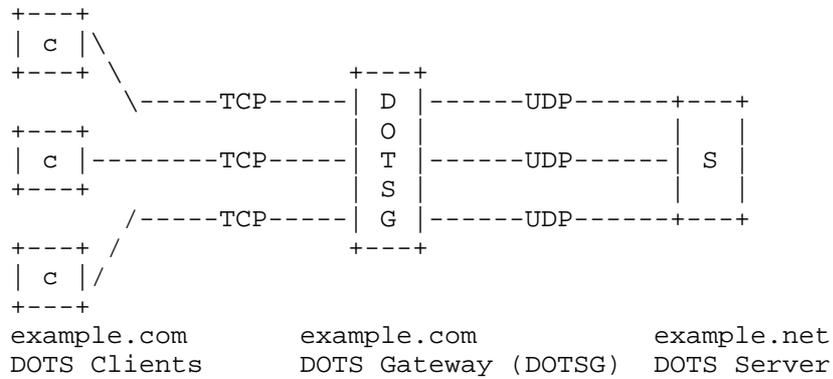
               Figure 8: Client-Side Gateway without Aggregation

   This may similarly be deployed in the inverse scenario where the
   gateway resides in the server-side domain and may be used to
   terminate and/or aggregate multiple clients to single transport as
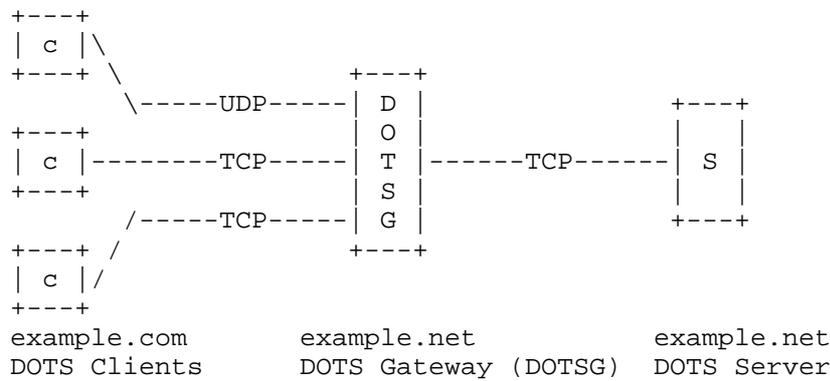   shown in figures Figure 9 and Figure 10 below.

```
     +---+
     | c |\
     +---+ \                    +---+
            \-----UDP-----| D |                    +---+
     +---+                | O |                    |   |
     | c |--------TCP-----| T |------TCP------| S |
     +---+                | S |                    |   |
            /-----TCP-----| G |                    +---+
     +---+ /                    +---+
     | c |/
     +---+
     example.com        example.net           example.net
     DOTS Clients       DOTS Gateway (DOTSG)   DOTS Server
```

            Figure 9: Server-Side Gateway with Aggregation

```
     +---+
     | c |\
     +---+ \                    +---+
            \-----UDP-----| D |------TCP------+---+
     +---+                | O |                    |   |
     | c |--------TCP-----| T |------TCP------| S |
     +---+                | S |                    |   |
            /-----UDP-----| G |------TCP------+---+
     +---+ /                    +---+
     | c |/
     +---+
     example.com        example.net           example.net
     DOTS Clients       DOTS Gateway (DOTSG)   DOTS Server
```
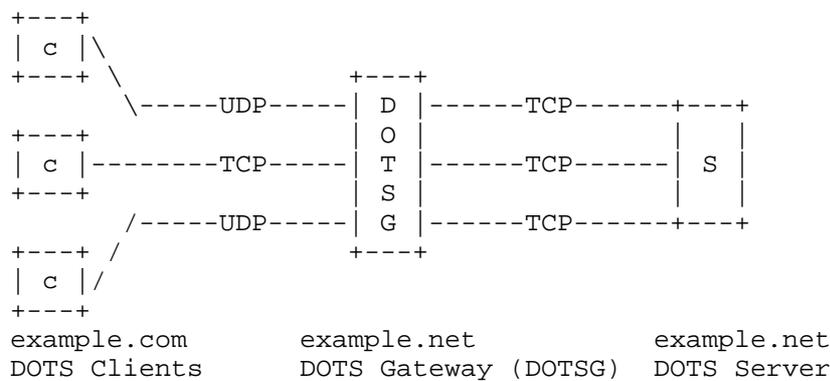
           Figure 10: Server-Side Gateway without Aggregation

3. Concepts

3.1. Signaling Sessions

   In order for DOTS to be effective as a vehicle for DDoS mitigation
   requests, one or more DOTS clients must establish ongoing
   communication with one or more DOTS servers. While the preconditions
   for enabling DOTS in or among network domains may also involve
   business relationships, service level agreements, or other formal or
   informal understandings between network operators, such
   considerations are out of scope for this document.

   An established communication layer between DOTS agents is a Signaling
   Session. At its most basic, for a DOTS signaling session to exist
   both signal channel and data channel must be functioning between DOTS
   agents. That is, under nominal network conditions, signals actively

sent from a DOTS client are received by the specific DOTS server
intended by the client, and vice versa.

3.1.1. Preconditions

Prior to establishing a signaling session between agents, the owners
of the networks, domains, services or applications involved are
assumed to have agreed upon the terms of the relationship involved.
Such agreements are out of scope for this document, but must be in
place for a functional DOTS architecture.

It is assumed that as part of any DOTS service agreement, the DOTS
client is provided with all data and metadata required to establish
communication with the DOTS server. Such data and metadata would
include any cryptographic information necessary to meet the message
confidentiality, integrity and authenticity requirement in
[I-D.ietf-dots-requirements], and might also include the pool of DOTS
server addresses and ports the DOTS client should use for signal and
data channel messaging.

3.1.2. Establishing the Signaling Session

With the required business or service agreements in place, the DOTS
client initiates a signal session by contacting the DOTS server over
the signal channel and the data channel. To allow for DOTS service
flexibility, neither the order of contact nor the time interval
between channel creations is specified. A DOTS client MAY establish
signal channel first, and then data channel, or vice versa.

The methods by which a DOTS client receives the address and
associated service details of the DOTS server are not prescribed by
this document. For example, a DOTS client may be directly configured
to use a specific DOTS server address and port, and directly provided
with any data necessary to satisfy the Peer Mutual Authentication
requirement in [I-D.ietf-dots-requirements], such as symmetric or
asymmetric keys, usernames and passwords, etc. All configuration and
authentication information in this scenario is provided out-of-band
by the domain operating the DOTS server.

At the other extreme, the architecture in this document allows for a
form of DOTS client auto-provisioning. For example, the domain
operating the DOTS server or servers might provide the client domain
only with symmetric or asymmetric keys to authenticate the
provisioned DOTS clients. Only the keys would then be directly
configured on DOTS clients, but the remaining configuration required
to provision the DOTS clients could be learned through mechanisms
similar to DNS SRV [RFC2782] or DNS Service Discovery [RFC6763].

The DOTS client SHOULD successfully authenticate and exchange
messages with the DOTS server over both signal and data channel as
soon as possible to confirm that both channels are operational.

As described in [I-D.ietf-dots-requirements], the DOTS client can
configure preferred values for acceptable signal loss, mitigation
lifetime, and heartbeat intervals when establishing the signaling
session. A signaling session is not active until DOTS agents have
agreed on the values for these signaling session parameters, a
process defined by the protocol.

Once the DOTS client begins receiving DOTS server signals, the
signaling session is active. At any time during the signaling
session, the DOTS client MAY use the data channel to adjust initial
configuration, manage black- and white-listed prefixes or addresses,
leverage vendor-specific extensions, and so on. Note that unlike the
signal channel, there is no requirement that the data channel remain
operational in attack conditions (See Data Channel Requirements,
[I-D.ietf-dots-requirements]).

3.1.3. Maintaining the Signaling Session

DOTS clients and servers periodically send heartbeats to each other
over the signal channel, per Operational Requirements discussed in
[I-D.ietf-dots-requirements]. DOTS agent operators SHOULD configure
the heartbeat interval such that the frequency does not lead to
accidental denials of service due to the overwhelming number of
heartbeats a DOTS agent must field.

Either DOTS agent may consider a signaling session terminated in the
extended absence of a heartbeat from its peer agent. The period of
that absence will be established in the protocol definition.

3.2. Modes of Signaling

This section examines the modes of signaling between agents in a DOTS
architecture.

3.2.1. Direct Signaling

A signaling session may take the form of direct signaling between the
DOTS clients and servers, as shown in Figure 11 below:

```
       +-------------+                           +-------------+
       | DOTS client |<------signal session----->| DOTS server |
       +-------------+                           +-------------+
```
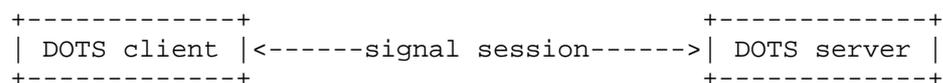
                      Figure 11: Direct Signaling

In a direct signaling session, DOTS client and server are
communicating directly. A direct signaling session MAY exist inter-
or intra-domain. The signaling session is abstracted from the
underlying networks or network elements the signals traverse: in a
direct signaling session, the DOTS client and server are logically
peer DOTS agents.

3.2.2. Redirected Signaling

In certain circumstances, a DOTS server may want to redirect a DOTS
client to an alternative DOTS server for a signaling session. Such
circumstances include but are not limited to:

o Maximum number of signaling sessions with clients has been
   reached;

o Mitigation capacity exhaustion in the Mitigator with which the
   specific DOTS server is communicating;

o Mitigator outage or other downtime, such as scheduled maintenance;

o Scheduled DOTS server maintenance;

o Scheduled modifications to the network path between DOTS server
   and DOTS client.

A basic redirected signaling session resembles the following, as
shown in Figure 12:

```
          +-------------+                        +--------------+
          |             |<-(1)-- signal session 1 -->|          |
          |             |             |          |              |
          |             |<=(2)== redirect to B ======|          |
          |  DOTS client |             |          | DOTS server A |
          |             |X-(4)-- signal session 1 --X|          |
          |             |             |          |              |
          |             |             |          |              |
          +-------------+                        +--------------+
                ^
                |
             (3) signal session 2
                |
                v
          +---------------+
          | DOTS server B |
          +---------------+
```
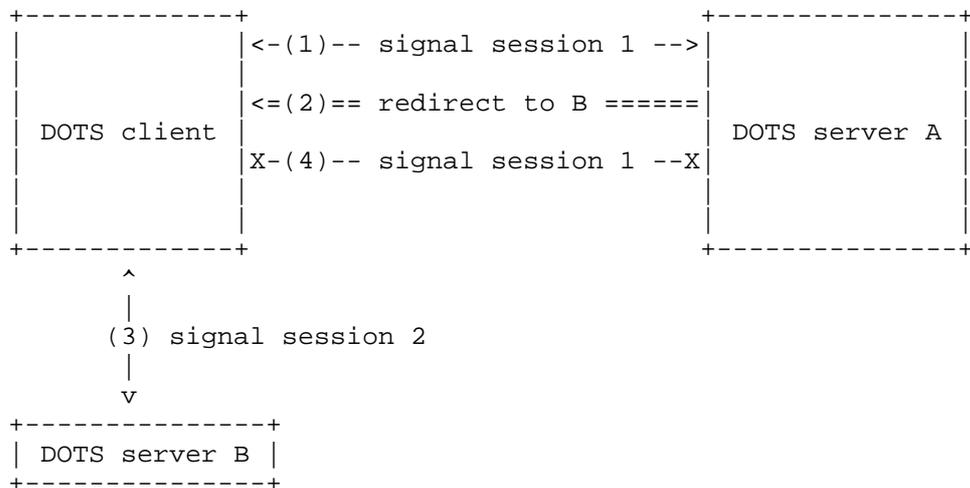
Figure 12: Redirected Signaling

   1. Previously established signaling session 1 exists between a DOTS
       client and DOTS server with address A.

   2. DOTS server A sends a server signal redirecting the client to
       DOTS server B.

   3. If the DOTS client does not already have a separate signaling
       session with the redirection target, the DOTS client initiates
       and establishes a signaling session with DOTS server B as
       described above.

   4. Having redirected the DOTS client, DOTS server A ceases sending
       server signals. The DOTS client likewise stops sending client
       signals to DOTS server A. Signal session 1 is terminated.

   [[EDITOR'S NOTE: we request working group feedback and discussion of
   the need for redirected signaling.]]

3.2.3. Recursive Signaling

   DOTS is centered around improving the speed and efficiency of
   coordinated response to DDoS attacks. One scenario not yet discussed
   involves coordination among federated domains operating DOTS servers
   and mitigators.

   In the course of normal DOTS operations, a DOTS client communicates
   the need for mitigation to a DOTS server, and that server initiates
   mitigation on a mitigator with which the server has an established
   service relationship. The operator of the mitigator may in turn
   monitor mitigation performance and capacity, as the attack being
   mitigated may grow in severity beyond the mitigating domain's
   capabilities.

   The operator of the mitigator has limited options in the event a DOTS
   client-requested mitigation is being overwhelmed by the severity of
   the attack. Out-of-scope business or service level agreements may
   permit the mitigating domain to drop the mitigation and let attack
   traffic flow unchecked to the target, but this is only encourages
   attack escalation. In the case where the mitigating domain is the
   upstream service provider for the attack target, this may mean the
   mitigating domain and its other services and users continue to suffer
   the incidental effects of the attack.

   A recursive signaling model as shown in Figure 13 below offers an
   alternative. In a variation of the primary use case "Successful
   Automatic or Operator-Assisted CPE or PE Mitigators Request Upstream
   DDoS Mitigation Services" described in [I-D.ietf-dots-use-cases], an
   domain operating a DOTS server and mitigation also operates a DOTS

client. This DOTS client has an established signaling session with a
DOTS server belonging to a separate administrative domain.

With these preconditions in place, the operator of the mitigator
being overwhelmed or otherwise performing inadequately may request
mitigation for the attack target from this separate DOTS-aware
domain. Such a request recurses the originating mitigation request
to the secondary DOTS server, in the hope of building a cumulative
mitigation against the attack:

```
                    example.net domain
                 . . . . . . . . . . . . . . . . . .
                 .       Gn                         .
        +----+     A   . +----+       +-----------+ .
        | Cc |<--------->| Sn |~~~~~~~| Mitigator | .
        +----+     .   +====+       |     Mn    | .
                 . | Cn |       +-----------+ .
    example.com  . +----+                     .
       client    .   ^                         .
                 . . .|. . . . . . . . . . . . . . .
                 .    |                         .
                 B |                         .
                 .    |                         .
                 . . .|. . . . . . . . . . . . . . .
                 .    v                         .
                 . +----+       +-----------+ .
                 . | So |~~~~~~~| Mitigator | .
                 . +----+       |     Mo    | .
                 .              +-----------+ .
                 .                             .
                 . . . . . . . . . . . . . . . . . .
                    example.org domain
```
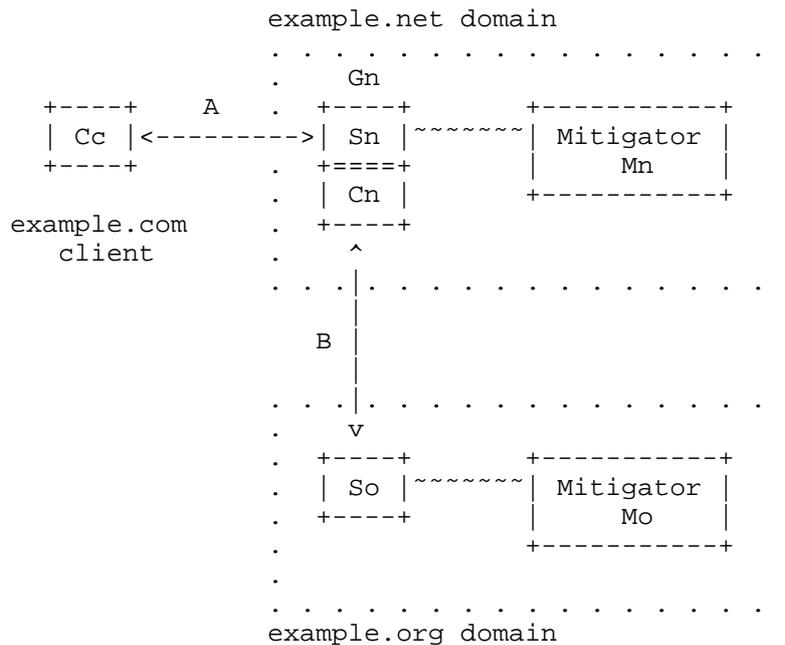
                  Figure 13: Recursive Signaling

   In Figure 13 above, client Cc signals a request for mitigation across
   inter-domain signaling session A to the DOTS server Sn belonging to
   the example.net domain. DOTS server Sn enables mitigation on
   mitigator Mn. DOTS server Sn is half of DOTS gateway Gn, being
   deployed logically back-to-back with DOTS client Cn, which has pre-
   existing inter-domain signaling session B with the DOTS server So
   belonging to the example.org domain. At any point, DOTS server Sn
   MAY recurse an on-going mitigation request through DOTS client Cn to
   DOTS server So, in the expectation that mitigator Mo will be
   activated to aid in the defense of the attack target.

   Recursive signaling is opaque to the DOTS client. To maximize
   mitigation visibility to the DOTS client, however, the recursing

domain SHOULD provide recursed mitigation feedback in signals
reporting on mitigation status to the DOTS client. For example, the
recursing domain's mitigator should incorporate into mitigation
status messages available metrics such as dropped packet or byte
counts from the recursed mitigation.

DOTS clients involved in recursive signaling MUST be able to withdraw
requests for mitigation without warning or justification, per
[I-D.ietf-dots-requirements].

Operators recursing mitigation requests MAY maintain the recursed
mitigation for a brief, protocol-defined period in the event the DOTS
client originating the mitigation withdraws its request for help, as
per the discussion of managing mitigation toggling in the operational
requirements ([I-D.ietf-dots-requirements]). Service or business
agreements between recursing domains are not in scope for this
document.

[[EDITOR'S NOTE: Recursive signaling raises questions about
operational and data privacy, as well as what level of visibility a
client has into the recursed mitigation. We ask the working group
for feedback and additional discussion of these issues to help settle
the way forward.]]

3.2.4. Anycast Signaling

The DOTS architecture does not assume the availability of anycast
within a DOTS deployment, but neither does the architecture exclude
it. Domains operating DOTS servers MAY deploy DOTS servers with an
anycast Service Address as described in BCP 126 [RFC4786]. In such a
deployment, DOTS clients connecting to the DOTS Service Address may
be communicating with distinct DOTS servers, depending on the network
configuration at the time the DOTS clients connect. Among other
benefits, anycasted signaling potentially offers the following:

o Simplified DOTS client configuration, including service discovery
   through the methods described in [RFC7094]. In this scenario, the
   "instance discovery" message would be a DOTS client initiating a
   signaling session to the DOTS server anycast Service Address, to
   which the DOTS server would reply with a redirection to the DOTS
   server unicast address the client should use for DOTS.

o Region- or customer-specific deployments, in which the DOTS
   Service Addresses route to distinct DOTS servers depending on the
   client region or the customer network in which a DOTS client
   resides.

   o Operational resiliency, spreading DOTS signaling traffic across
      the DOTS server domain's networks, and thereby also reducing the
      potential attack surface, as described in BCP 126 [RFC4786].

3.2.4.1. Anycast Signaling Considerations

   As long as network configuration remains stable, anycast DOTS
   signaling is to the individual DOTS client indistinct from direct
   signaling. However, the operational challenges inherent in anycast
   signaling are anything but negligible, and DOTS server operators must
   carefully weigh the risks against the benefits before deploying.

   While the DOTS signal channel primarily operates over UDP per
   [I-D.ietf-dots-requirements], the signal channel also requires mutual
   authentication between DOTS agents, with associated security state on
   both ends. The resulting considerations therefore superficially
   resemble the deployment of anycast DNS over DTLS, as described in
   [I-D.ietf-dprive-dnsodtls], but the similiarities only go so far.

   Network instability is of particular concern with anycast signaling,
   as DOTS signaling sessions are expected to be long-lived, and
   potentially operating under congested network conditions caused by a
   volumetric DDoS attack.

   For example, a network configuration altering the route to the DOTS
   server during active anycast signaling may cause the DOTS client to
   send messages to a DOTS server other than the one with which it
   initially established a signaling session. That second DOTS server
   may not have the security state of the existing session, forcing the
   DOTS client to initialize a new signaling session. This challenge
   may in part be mitigated by use of pre-shared keys, as described in
   [I-D.ietf-tls-tls13], but keying material must be available to all
   DOTS servers sharing the anycast Service Address in that case.

   While the DOTS client will try to establish a new signaling session
   with the DOTS server now acting as the anycast DOTS Service Address,
   the link between DOTS client and server may be congested with attack
   traffic, making signal session establishment difficult. In such a
   scenario, anycast Service Address instability becomes a sort of
   signal session flapping, with obvious negative consequences for the
   DOTS deployment.

   Anycast signaling deployments similarly must also take into account
   active mitigations. Active mitigations initiated through a signaling
   session may involve diverting traffic to a scrubbing center. If the
   signaling session flaps due to anycast changes as described above,
   mitigation may also flap as the DOTS servers sharing the anycast DOTS
   service address toggles mitigation on detecting signaling session

   loss, depending on whether the client has configured mitigation on
   loss of signal.

   [[EDITOR'S NOTE: We request feedback from the working group regarding
   the complexities inherent in an anycast DOTS deployment. Outside of
   using anycast for service discovery, significant challenges need to
   be overcome, particularly when dealing with security and mitigation
   state, and the resulting operational complexity may outweigh the
   expected benefits.]]

3.3. Triggering Requests for Mitigation

   [I-D.ietf-dots-requirements] places no limitation on the
   circumstances in which a DOTS client operator may request mitigation,
   nor does it demand justification for any mitigation request, thereby
   reserving operational control over DDoS defense for the domain
   requesting mitigation. This architecture likewise does not prescribe
   the network conditions and mechanisms triggering a mitigation request
   from a DOTS client.

   However, considering selected possible mitigation triggers from an
   architectural perspective offers a model for alternative or
   unanticipated triggers for DOTS deployments. In all cases, what
   network conditions merit a mitigation request are at the discretion
   of the DOTS client operator.

   The interfaces required to trigger the mitigation request in the
   following scenarios are implementation-specific.

3.3.1. Manual Mitigation Request

   A DOTS client operator may manually prepare a request for mitigation,
   including scope and duration, and manually instruct the DOTS client
   to send the mitigation request to the DOTS server. In context, a
   manual request is a request directly issued by the operator without
   automated decision-making performed by a device interacting with the
   DOTS client. Modes of manual mitigation requests include an operator
   entering a command into a text interface, or directly interacting
   with a graphical interface to send the request.

   An operator might do this, for example, in response to notice of an
   attack delivered by attack detection equipment or software, and the
   alerting detector lacks interfaces or is not configured to use
   available interfaces to translate the alert to a mitigation request
   automatically.

   In a variation of the above scenario, the operator may have
   preconfigured on the DOTS client mitigation request for various

   resources in the operator's domain. When notified of an attack, the
   DOTS client operator manually instructs the DOTS client to send the
   preconfigured mitigation request for the resources under attack.

   A further variant involves recursive signaling, as described in
   Section 3.2.3. The DOTS client in this case is the second half of a
   DOTS gateway (back-to-back DOTS server and client). As in the
   previous scenario, the scope and duration of the mitigation request
   are pre-existing, but in this case are derived from the mitigation
   request received from a downstream DOTS client by the DOTS server.
   Assuming the preconditions required by Section 3.2.3 are in place,
   the DOTS gateway operator may at any time manually request mitigation
   from an upstream DOTS server, sending a mitigation request derived
   from the downstream DOTS client's request.

   The motivations for a DOTS client operator to request mitigation
   manually are not prescribed by this architecture, but are expected to
   include some of the following:

   o Notice of an attack delivered via e-mail or alternative messaging

   o Notice of an attack delivered via phone call

   o Notice of an attack delivered through the interface(s) of
      networking monitoring software deployed in the operator's domain

   o Manual monitoring of network behavior through network monitoring
      software

3.3.2. Automated Conditional Mitigation Request

   Unlike manual mitigation requests, which depend entirely on the DOTS
   client operator's capacity to react with speed and accuracy to every
   detected or detectable attack, mitigation requests triggered by
   detected attack conditions reduce the operational burden on the DOTS
   client operator, and minimize the latency between attack detection
   and the start of mitigation.

   Mitigation requests are triggered in this scenario by operator-
   specified network conditions. Attack detection is deployment-
   specific, and not constrained by this architecture. Similarly the
   specifics of a condition are left to the discretion of the operator,
   though common conditions meriting mitigation include the following:

   o Detected attack exceeding a rate in packets per second (pps).

   o Detected attack exceeding a rate in bytes per second (bps).

o Detected resource exhaustion in an attack target.

o Detected resource exhaustion in the local domain's mitigator.

o Number of open connections to an attack target.

o Number of attack sources in a given attack.

o Number of active attacks against targets in the operator's domain.

o Conditional detection developed through arbitrary statistical
   analysis or deep learning techniques.

o Any combination of the above.

When automated conditional mitigation requests are enabled,
violations of any of the above conditions, or any additional
operator-defined conditions, will trigger a mitigation request from
the DOTS client to the DOTS server. The interfaces between the
application detecting the condition violation and the DOTS client are
implementation-specific.

3.3.3. Automated Mitigation on Loss of Signal

To maintain a signaling session, the DOTS client and the DOTS server
exchange regular but infrequent messages across the signaling
channel. In the absence of an attack, the probability of message
loss in the signaling channel should be extremely low. Under attack
conditions, however, some signal loss may be anticipated as attack
traffic congests the link, depending on the attack type.

While [I-D.ietf-dots-requirements] specifies the DOTS protocol be
robust when signaling under attack conditions, there are nevertheless
scenarios in which the DOTS signal is lost in spite of protocol best
efforts. To handle such scenarios, a DOTS client operator may
configure the signaling session to trigger mitigation when the DOTS
server ceases receiving DOTS client signals (or vice versa) beyond
the miss count or period permitted by the protocol.

The impact of mitigating due to loss of signal in either direction
must be considered carefully before enabling it. Signal loss is not
caused by links congested with attack traffic alone, and as such
mitigation requests triggered by signal channel degradation in either
direction may incur unnecessary costs, in network performance and
operational expense alike.

4. Security Considerations

    This section describes identified security considerations for the
    DOTS architecture.

    DOTS is at risk from three primary attack vectors: agent
    impersonation, traffic injection and signal blocking. These vectors
    may be exploited individually or in concert by an attacker to
    confuse, disable, take information from, or otherwise inhibit DOTS
    agents.

    Any attacker with the ability to impersonate a legitimate client or
    server or, indeed, inject false messages into the stream may
    potentially trigger/withdraw traffic redirection, trigger/cancel
    mitigation activities or subvert black/whitelists. From an
    architectural standpoint, operators SHOULD ensure best current
    practices for secure communication are observed for data and signal
    channel confidentiality, integrity and authenticity. Care must be
    taken to ensure transmission is protected by appropriately secure
    means, reducing attack surface by exposing only the minimal required
    services or interfaces. Similarly, received data at rest SHOULD be
    stored with a satisfactory degree of security.

    As many mitigation systems employ diversion to scrub attack traffic,
    operators of DOTS agents SHOULD ensure signaling sessions are
    resistant to Man-in-the-Middle (MitM) attacks. An attacker with
    control of a DOTS client may negatively influence network traffic by
    requesting and withdrawing requests for mitigation for particular
    prefixes, leading to route or DNS flapping.

    Any attack targeting the availability of DOTS servers may disrupt the
    ability of the system to receive and process DOTS signals resulting
    in failure to fulfill a mitigation request. DOTS agents SHOULD be
    given adequate protections, again in accordance with best current
    practices for network and host security.

5. Acknowledgments

    Thanks to Matt Richardson and Med Boucadair for their comments and
    suggestions.

6. Change Log

    2016-03-18 Initial revision

7. References

7.1. Normative References

   [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>.

7.2. Informative References

   [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-02 (work in
              progress), July 2016.

   [I-D.ietf-dots-use-cases]
              Dobbins, R., Fouant, S., Migault, D., Moskowitz, R.,
              Teague, N., and L. Xia, "Use cases for DDoS Open Threat
              Signaling", draft-ietf-dots-use-cases-01 (work in
              progress), March 2016.

   [I-D.ietf-dprive-dnsodtls]
              Reddy, T., Wing, D., and P. Patil, "Specification for DNS
              over Datagram Transport Layer Security (DTLS)", draft-
              ietf-dprive-dnsodtls-12 (work in progress), September
              2016.

   [I-D.ietf-tls-tls13]
              Rescorla, E., "The Transport Layer Security (TLS) Protocol
              Version 1.3", draft-ietf-tls-tls13-18 (work in progress),
              October 2016.

   [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768,
              DOI 10.17487/RFC0768, August 1980,
              <http://www.rfc-editor.org/info/rfc768>.

   [RFC0793] Postel, J., "Transmission Control Protocol", STD 7,
              RFC 793, DOI 10.17487/RFC0793, September 1981,
              <http://www.rfc-editor.org/info/rfc793>.

   [RFC1034] Mockapetris, P., "Domain names - concepts and facilities",
              STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987,
              <http://www.rfc-editor.org/info/rfc1034>.

   [RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for
              specifying the location of services (DNS SRV)", RFC 2782,
              DOI 10.17487/RFC2782, February 2000,
              <http://www.rfc-editor.org/info/rfc2782>.

   [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston,
              A., Peterson, J., Sparks, R., Handley, M., and E.
              Schooler, "SIP: Session Initiation Protocol", RFC 3261,
              DOI 10.17487/RFC3261, June 2002,
              <http://www.rfc-editor.org/info/rfc3261>.

   [RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A
              Border Gateway Protocol 4 (BGP-4)", RFC 4271,
              DOI 10.17487/RFC4271, January 2006,
              <http://www.rfc-editor.org/info/rfc4271>.

   [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>.

   [RFC4786] Abley, J. and K. Lindqvist, "Operation of Anycast
              Services", BCP 126, RFC 4786, DOI 10.17487/RFC4786,
              December 2006, <http://www.rfc-editor.org/info/rfc4786>.

   [RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service
              Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013,
              <http://www.rfc-editor.org/info/rfc6763>.

   [RFC7092] Kaplan, H. and V. Pascual, "A Taxonomy of Session
              Initiation Protocol (SIP) Back-to-Back User Agents",
              RFC 7092, DOI 10.17487/RFC7092, December 2013,
              <http://www.rfc-editor.org/info/rfc7092>.

   [RFC7094] McPherson, D., Oran, D., Thaler, D., and E. Osterweil,
              "Architectural Considerations of IP Anycast", RFC 7094,
              DOI 10.17487/RFC7094, January 2014,
              <http://www.rfc-editor.org/info/rfc7094>.

Authors' Addresses

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

   EMail: amortensen@arbor.net

Flemming Andreasen
Cisco Systems, Inc.
United States

EMail: fandreas@cisco.com

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

EMail: tireddy@cisco.com

Christopher Gray
Comcast, Inc.
United States

EMail: Christopher_Gray3@cable.comcast.com

Rich Compton
Charter Communications, Inc.

EMail: Rich.Compton@charter.com

Nik Teague
Verisign, Inc.
United States

EMail: nteague@verisign.com

DOTS   A. Mortensen
Internet-Draft   Arbor Networks
Intended status: Informational   F. Andreasen
Expires: September 21, 2018   Cisco
                                                                T. Reddy
                                                             McAfee, Inc.
                                                                 C. Gray

                                                              R. Compton
                                                                 Charter
                                                                N. Teague
                                                                Verisign
                                                          March 20, 2018

       Distributed-Denial-of-Service Open Threat Signaling (DOTS) Architecture
                      draft-ietf-dots-architecture-06

   Abstract

      This document describes an architecture for establishing and
      maintaining Distributed Denial of Service (DDoS) Open Threat
      Signaling (DOTS) within and between domains. The document does not
      specify protocols or protocol extensions, instead focusing on
      defining architectural relationships, components and concepts used in
      a DOTS deployment.

Copyright Notice

Table of Contents

1. Context and Motivation

   Signaling the need for help defending against an active distributed
   denial of service (DDoS) attack requires a common understanding of
   mechanisms and roles among the parties coordinating defensive
   response. The signaling layer and supplementary messaging is the
   focus of DDoS Open Threat Signaling (DOTS). DOTS defines a method of
   coordinating defensive measures among willing peers to mitigate
   attacks quickly and efficiently, enabling hybrid attack responses
   coordinated locally at or near the target of an active attack, or
   anywhere in-path between attack sources and target. Sample DOTS use
   cases are elaborated in [I-D.ietf-dots-use-cases].

   This document describes an architecture used in establishing,
   maintaining or terminating a DOTS relationship within a domain or
   between domains.

1.1. Terminology

1.1.1. Key Words

   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].

1.1.2. Definition of Terms

   This document uses the terms defined in [I-D.ietf-dots-requirements].

1.2. Scope

   In this architecture, DOTS clients and servers communicate using DOTS
   signaling. As a result of signals from a DOTS client, the DOTS
   server may modify the forwarding path of traffic destined for the
   attack target(s), for example by diverting traffic to a mitigator or
   pool of mitigators, where policy may be applied to distinguish and
   discard attack traffic. Any such policy is deployment-specific.

   The DOTS architecture presented here is applicable across network
   administrative domains - for example, between an enterprise domain
   and the domain of a third-party attack mitigation service - as well

as to a single administrative domain. DOTS is generally assumed to
be most effective when aiding coordination of attack response between
two or more participating networks, but single domain scenarios are
valuable in their own right, as when aggregating intra-domain DOTS
client signals for inter-domain coordinated attack response.

This document does not address any administrative or business
agreements that may be established between involved DOTS parties.
Those considerations are out of scope. Regardless, this document
assumes necessary authentication and authorization mechanisms are put
in place so that only authorized clients can invoke the DOTS service.

A detailed set of DOTS requirements are discussed in
[I-D.ietf-dots-requirements], and the DOTS architecture is designed
to follow those requirements. Only new behavioral requirements are
described in this document.

1.3. Assumptions

This document makes the following assumptions:

o All domains in which DOTS is deployed are assumed to offer the
   required connectivity between DOTS agents and any intermediary
   network elements, but the architecture imposes no additional
   limitations on the form of connectivity.

o Congestion and resource exhaustion are intended outcomes of a DDoS
   attack [RFC4732]. Some operators may utilize non-impacted paths
   or networks for DOTS, but in general conditions should be assumed
   to be hostile and DOTS must be able to function in all
   circumstances, including when the signaling path is significantly
   impaired.

o There is no universal DDoS attack scale threshold triggering a
   coordinated response across administrative domains. A network
   domain administrator, or service or application owner may
   arbitrarily set attack scale threshold triggers, or manually send
   requests for mitigation.

o Mitigation requests may be sent to one or more upstream DOTS
   servers based on criteria determined by DOTS client administrators
   and the underlying network configuration. The number of DOTS
   servers with which a given DOTS client has established
   communications is determined by local policy and is deployment-
   specific. For example, a DOTS client of a multi-homed network may
   support built-in policies to establish DOTS relationships with
   DOTS servers located upstream of each interconnection link.

    o The mitigation capacity and/or capability of domains receiving
       requests for coordinated attack response is opaque to the domains
       sending the request. The domain receiving the DOTS client signal
       may or may not have sufficient capacity or capability to filter
       any or all DDoS attack traffic directed at a target. In either
       case, the upstream DOTS server may redirect a request to another
       DOTS server. Redirection may be local to the redirecting DOTS
       server's domain, or may involve a third-party domain.

    o DOTS client and server signals, as well as messages sent through
       the data channel, are sent across any transit networks with the
       same probability of delivery as any other traffic between the DOTS
       client domain and the DOTS server domain. Any encapsulation
       required for successful delivery is left untouched by transit
       network elements. DOTS server and DOTS client cannot assume any
       preferential treatment of DOTS signals. Such preferential
       treatment may be available in some deployments (e.g., intra-domain
       scenarios), and the DOTS architecture does not preclude its use
       when available. However, DOTS itself does not address how that
       may be done.

    o The architecture allows for, but does not assume, the presence of
       Quality of Service (QoS) policy agreements between DOTS-enabled
       peer networks or local QoS prioritization aimed at ensuring
       delivery of DOTS messages between DOTS agents. QoS is an
       operational consideration only, not a functional part of the DOTS
       architecture.

    o The signal and data channels are loosely coupled, and may not
       terminate on the same DOTS server.

2. DOTS Architecture

   The basic high-level DOTS architecture is illustrated in Figure 1:

        +-----------+   +-------------+
        | Mitigator | ~~~~~~~~~~ | DOTS Server |
        +-----------+   +-------------+
                                              |
                                              |
                                              |
        +---------------+   +-------------+
        | Attack Target | ~~~~~ | DOTS Client |
        +---------------+   +-------------+
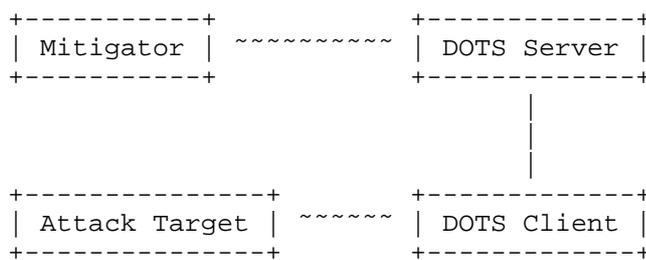

                   Figure 1: Basic DOTS Architecture

A simple example instantiation of the DOTS architecture could be an
enterprise as the attack target for a volumetric DDoS attack, and an
upstream DDoS mitigation service as the mitigator. The enterprise
(attack target) is connected to the Internet via a link that is
getting saturated, and the enterprise suspects it is under DDoS
attack. The enterprise has a DOTS client, which obtains information
about the DDoS attack, and signals the DOTS server for help in
mitigating the attack. The DOTS server in turn invokes one or more
mitigators, which are tasked with mitigating the actual DDoS attack,
and hence aim to suppress the attack traffic while allowing valid
traffic to reach the attack target.

The scope of the DOTS specifications is the interfaces between the
DOTS client and DOTS server. The interfaces to the attack target and
the mitigator are out of scope of DOTS. Similarly, the operation of
both the attack target and the mitigator is out of scope of DOTS.
Thus, DOTS neither specifies how an attack target decides it is under
DDoS attack, nor does DOTS specify how a mitigator may actually
mitigate such an attack. A DOTS client's request for mitigation is
advisory in nature, and may not lead to any mitigation at all,
depending on the DOTS server domain's capacity and willingness to
mitigate on behalf of the DOTS client's domain.

The DOTS client may be provided with a list of DOTS servers, each
associated with one or more IP addresses. These addresses may or may
not be of the same address family. The DOTS client establishes one
or more sessions by connecting to the provided DOTS server addresses.

As illustrated in Figure 2, there are two interfaces between a DOTS
server and a DOTS client; a signal channel and (optionally) a data
channel.

```
+---------------+                                    +---------------+
|               | <------- Signal Channel ------> |               |
|  DOTS Client  |                                    |  DOTS Server  |
|               | <======= Data Channel  ======> |               |
+---------------+                                    +---------------+
```
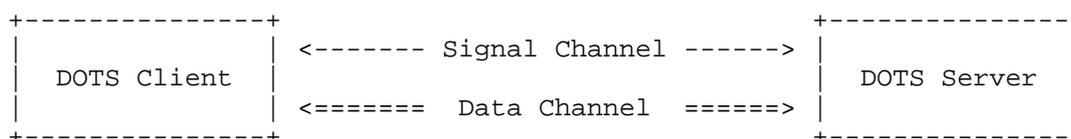
Figure 2: DOTS Interfaces

The primary purpose of the signal channel is for a DOTS client to ask
a DOTS server for help in mitigating an attack, and for the DOTS
server to inform the DOTS client about the status of such mitigation.
The DOTS client does this by sending a client signal, which contains
information about the attack target(s). The client signal may also
include telemetry information about the attack, if the DOTS client
has such information available. The DOTS server in turn sends a
server signal to inform the DOTS client of whether it will honor the

mitigation request. Assuming it will, the DOTS server initiates
attack mitigation, and periodically informs the DOTS client about the
status of the mitigation. Similarly, the DOTS client periodically
informs the DOTS server about the client's status, which at a minimum
provides client (attack target) health information, but it should
also include efficacy information about the attack mitigation as it
is now seen by the client. At some point, the DOTS client may decide
to terminate the server-side attack mitigation, which it indicates to
the DOTS server over the signal channel. A mitigation may also be
terminated if a DOTS client-specified mitigation lifetime is
exceeded. Note that the signal channel may need to operate over a
link that is experiencing a DDoS attack and hence is subject to
severe packet loss and high latency.

While DOTS is able to request mitigation with just the signal
channel, the addition of the DOTS data channel provides for
additional and more efficient capabilities. The primary purpose of
the data channel is to support DOTS related configuration and policy
information exchange between the DOTS client and the DOTS server.
Examples of such information include, but are not limited to:

o Creating identifiers, such as names or aliases, for resources for
   which mitigation may be requested. Such identifiers may then be
   used in subsequent signal channel exchanges to refer more
   efficiently to the resources under attack, as seen in Figure 3,
   using JSON to serialize the data:

```
{
    "https1": [
        "192.0.2.1:443",
        "198.51.100.2:443",
    ],
    "proxies": [
        "203.0.113.3:3128",
        "[2001:db8:ac10::1]:3128"
    ],
    "api_urls": "https://apiserver.example.com/api/v1",
}
```

              Figure 3: Protected resource identifiers

o Black-list management, which enables a DOTS client to inform the
   DOTS server about sources to suppress.

o White-list management, which enables a DOTS client to inform the
   DOTS server about sources from which traffic is always accepted.

      o Filter management, which enables a DOTS client to install or
         remove traffic filters dropping or rate-limiting unwanted traffic.

      o DOTS client provisioning.

   Note that while it is possible to exchange the above information
   before, during or after a DDoS attack, DOTS requires reliable
   delivery of this information and does not provide any special means
   for ensuring timely delivery of it during an attack. In practice,
   this means that DOTS deployments should not rely on such information
   being exchanged during a DDoS attack.

2.1. DOTS Operations

   DOTS does not prescribe any specific deployment models, however DOTS
   is designed with some specific requirements around the different DOTS
   agents and their relationships.

   First of all, a DOTS agent belongs to a domain that has an identity
   which can be authenticated and authorized. DOTS agents communicate
   with each other over a mutually authenticated signal channel and
   (optionally) data channel. However, before they can do so, a service
   relationship needs to be established between them. The details and
   means by which this is done is outside the scope of DOTS, however an
   example would be for an enterprise A (DOTS client) to sign up for
   DDoS service from provider B (DOTS server). This would establish a
   (service) relationship between the two that enables enterprise A's
   DOTS client to establish a signal channel with provider B's DOTS
   server. A and B will authenticate each other, and B can verify that
   A is authorized for its service.

   From an operational and design point of view, DOTS assumes that the
   above relationship is established prior to a request for DDoS attack
   mitigation. In particular, it is assumed that bi-directional
   communication is possible at this time between the DOTS client and
   DOTS server. Furthermore, it is assumed that additional service
   provisioning, configuration and information exchange can be performed
   by use of the data channel, if operationally required. It is not
   until this point that the mitigation service is available for use.

   Once the mutually authenticated signal channel has been established,
   it will remain active. This is done to increase the likelihood that
   the DOTS client can signal the DOTS server for help when the attack
   target is being flooded, and similarly raise the probability that
   DOTS server signals reach the client regardless of inbound link
   congestion. This does not necessarily imply that the attack target
   and the DOTS client have to be co-located in the same administrative
   domain, but it is expected to be a common scenario.

DDoS mitigation with the help of an upstream mitigator may involve
some form of traffic redirection whereby traffic destined for the
attack target is steered towards the mitigator. Common mechanisms to
achieve this redirection depend on BGP [RFC4271] and DNS [RFC1035].
The mitigator in turn inspects and scrubs the traffic, and forwards
the resulting (hopefully non-attack) traffic to the attack target.
Thus, when a DOTS server receives an attack mitigation request from a
DOTS client, it can be viewed as a way of causing traffic redirection
for the attack target indicated.

DOTS relies on mutual authentication and the pre-established service
relationship between the DOTS client's domain and the DOTS server's
domain to provide basic authorization. The DOTS server should
enforce additional authorization mechanisms to restrict the
mitigation scope a DOTS client can request, but such authorization
mechanisms are deployment-specific.

Although co-location of DOTS server and mitigator within the same
domain is expected to be a common deployment model, it is assumed
that operators may require alternative models. Nothing in this
document precludes such alternatives.

2.2. Components

2.2.1. DOTS Client

A DOTS client is a DOTS agent from which requests for help
coordinating attack response originate. The requests may be in
response to an active, ongoing attack against a target in the DOTS
client's domain, but no active attack is required for a DOTS client
to request help. Operators may wish to have upstream mitigators in
the network path for an indefinite period, and are restricted only by
business relationships when it comes to duration and scope of
requested mitigation.

The DOTS client requests attack response coordination from a DOTS
server over the signal channel, including in the request the DOTS
client's desired mitigation scoping, as described in
[I-D.ietf-dots-requirements]. The actual mitigation scope and
countermeasures used in response to the attack are up to the DOTS
server and mitigator operators, as the DOTS client may have a narrow
perspective on the ongoing attack. As such, the DOTS client's
request for mitigation should be considered advisory: guarantees of
DOTS server availability or mitigation capacity constitute service
level agreements and are out of scope for this document.

The DOTS client adjusts mitigation scope and provides available
mitigation feedback (e.g. mitigation efficacy) at the direction of

its local administrator. Such direction may involve manual or
automated adjustments in response to updates from the DOTS server.

To provide a metric of signal health and distinguish an idle signal
channel from a disconnected or defunct session, the DOTS client sends
a heartbeat over the signal channel to maintain its half of the
channel. The DOTS client similarly expects a heartbeat from the DOTS
server, and may consider a session terminated in the extended absence
of a DOTS server heartbeat.

2.2.2. DOTS Server

A DOTS server is a DOTS agent capable of receiving, processing and
possibly acting on requests for help coordinating attack response
from DOTS clients. The DOTS server authenticates and authorizes DOTS
clients as described in Section 3.1, and maintains session state,
tracking requests for mitigation, reporting on the status of active
mitigations, and terminating sessions in the extended absence of a
client heartbeat or when a session times out.

Assuming the preconditions discussed below exist, a DOTS client
maintaining an active session with a DOTS server may reasonably
expect some level of mitigation in response to a request for
coordinated attack response.

The DOTS server enforces authorization of DOTS clients' signals for
mitigation. The mechanism of enforcement is not in scope for this
document, but is expected to restrict requested mitigation scope to
addresses, prefixes, and/or services owned by the DOTS client's
administrative domain, such that a DOTS client from one domain is not
able to influence the network path to another domain. A DOTS server
MUST reject requests for mitigation of resources not owned by the
requesting DOTS client's administrative domain. A DOTS server MAY
also refuse a DOTS client's mitigation request for arbitrary reasons,
within any limits imposed by business or service level agreements
between client and server domains. If a DOTS server refuses a DOTS
client's request for mitigation, the DOTS server SHOULD include the
refusal reason in the server signal sent to the client.

A DOTS server is in regular contact with one or more mitigators. If
a DOTS server accepts a DOTS client's request for help, the DOTS
server forwards a translated form of that request to the mitigator(s)
responsible for scrubbing attack traffic. Note that the form of the
translated request passed from the DOTS server to the mitigator is
not in scope: it may be as simple as an alert to mitigator operators,
or highly automated using vendor or open application programming
interfaces supported by the mitigator. The DOTS server MUST report
the actual scope of any mitigation enabled on behalf of a client.

The DOTS server SHOULD retrieve available metrics for any mitigations
activated on behalf of a DOTS client, and SHOULD include them in
server signals sent to the DOTS client originating the request for
mitigation.

To provide a metric of signal health and distinguish an idle signal
channel from a disconnected or defunct channel, the DOTS server MUST
send a heartbeat over the signal channel to maintain its half of the
channel. The DOTS server similarly expects a heartbeat from the DOTS
client, and MAY consider a session terminated in the extended absence
of a DOTS client heartbeat.

2.2.3. DOTS Gateway

   Traditional client/server relationships may be expanded by chaining
   DOTS sessions. This chaining is enabled through "logical
   concatenation" of a DOTS server and a DOTS client, resulting in an
   application analogous to the Session Initiation Protocol (SIP)
   [RFC3261] logical entity of a Back-to-Back User Agent (B2BUA)
   [RFC7092]. The term DOTS gateway is used here in the descriptions of
   selected scenarios involving this application.

   A DOTS gateway may be deployed client-side, server-side or both. The
   gateway may terminate multiple discrete client connections and may
   aggregate these into a single or multiple DOTS sessions.

   The DOTS gateway will appear as a server to its downstream agents and
   as a client to its upstream agents, a functional concatenation of the
   DOTS client and server roles, as depicted in Figure 4:

```
                        +-------------+
                        |     | D |     |
        +----+          |     | O |     |          +----+
        | c1 |----------| s1  | T | c2  |---------| s2 |
        +----+          |     | S |     |          +----+
                        |     | G |     |
                        +-------------+
```

                        Figure 4: DOTS gateway

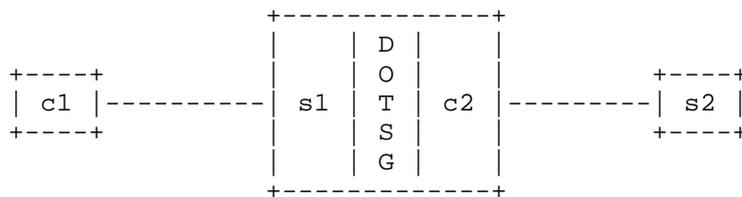   The DOTS gateway MUST perform full stack DOTS session termination and
   reorigination between its client and server side. The details of how
   this is achieved are implementation specific. The DOTS protocol does
   not include any special features related to DOTS gateways, and hence
   from a DOTS perspective, whenever a DOTS gateway is present, the DOTS
   session simply terminates/originates there.

2.3. DOTS Agent Relationships

   So far, we have only considered a relatively simple scenario of a
   single DOTS client associated with a single DOTS server, however DOTS
   supports more advanced relationships.

   A DOTS server may be associated with one or more DOTS clients, and
   those DOTS clients may belong to different domains. An example
   scenario is a mitigation provider serving multiple attack targets
   (Figure 5).

```
      DOTS clients        DOTS server
      +---+
      | c |-----------
      +---+             \
      c1.example.org   \
                         \
      +---+               \ +---+
      | c |---------------| S |
      +---+               / +---+
      c1.example.com     /  dots1.example.net
                        /
      +---+           /
      | c |-----------
      +---+
      c2.example.com
```

                 Figure 5: DOTS server with multiple clients
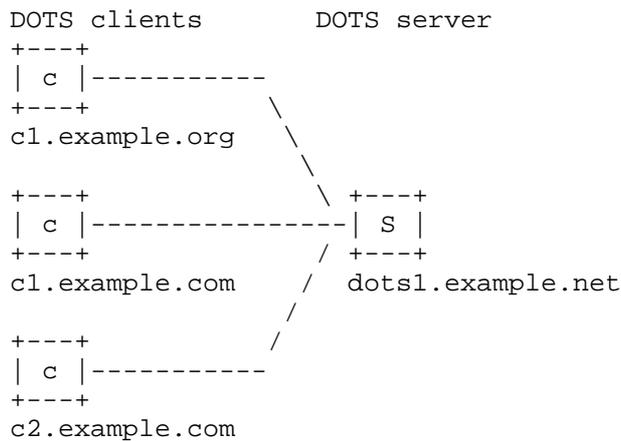
   A DOTS client may be associated with one or more DOTS servers, and
   those DOTS servers may belong to different domains. This may be to
   ensure high availability or co-ordinate mitigation with more than one
   directly connected ISP. An example scenario is for an enterprise to
   have DDoS mitigation service from multiple providers, as shown in
   Figure 6.

```
   DOTS client          DOTS servers
                            +---+
                ----------| S |
               /            +---+
              /              dots1.example.net
             /
   +---+/                   +---+
   | c |---------------| S |
   +---+\                   +---+
         \                  dots.example.org
          \
           \              +---+
            ----------| S |
                            +---+
   c.example.com        dots2.example.net
```
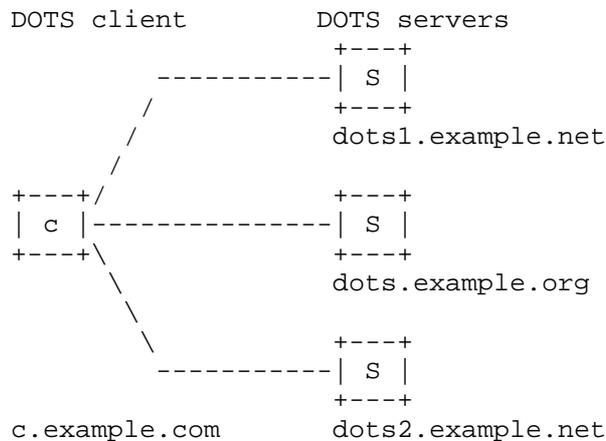
                    Figure 6: Multi-Homed DOTS Client

   Deploying a multi-homed client requires extra care and planning, as
   the DOTS servers with which the multi-homed client communicates may
   not be affiliated. Should the multi-homed client simultaneously
   request for mitigation from all servers with which it has established
   signal channels, the client may unintentionally inflict additional
   network disruption on the resources it intends to protect. In one of
   the worst cases, a multi-homed DOTS client could cause a permanent
   routing loop of traffic destined for the client's protected services,
   as the uncoordinated DOTS servers' mitigators all try to divert that
   traffic to their own scrubbing centers.

   The DOTS protocol itself provides no fool-proof method to prevent
   such self-inflicted harms as a result deploying multi-homed DOTS
   clients. If DOTS client implementations nevertheless include support
   for multi-homing, they are expected to be aware of the risks, and
   consequently to include measures aimed at reducing the likelihood of
   negative outcomes. Simple measures might include:

   o Requesting mitigation serially, ensuring only one mitigation
      request for a given address space is active at any given time;

   o Dividing the protected resources among the DOTS servers, such that
      no two mitigators will be attempting to divert and scrub the same
      traffic;

   o Restricting multi-homing to deployments in which all DOTS servers
      are coordinating management of a shared pool of mitigation
      resources.

2.3.1. Gatewayed Signaling

   As discussed in Section 2.2.3, a DOTS gateway is a logical function
   chaining DOTS sessions through concatenation of a DOTS server and
   DOTS client.

   An example scenario, as shown in Figure 7 and Figure 8, is for an
   enterprise to have deployed multiple DOTS capable devices which are
   able to signal intra-domain using TCP [RFC0793] on un-congested links
   to a DOTS gateway which may then transform these to a UDP [RFC0768]
   transport inter-domain where connection oriented transports may
   degrade; this applies to the signal channel only, as the data channel
   requires a connection-oriented transport. The relationship between
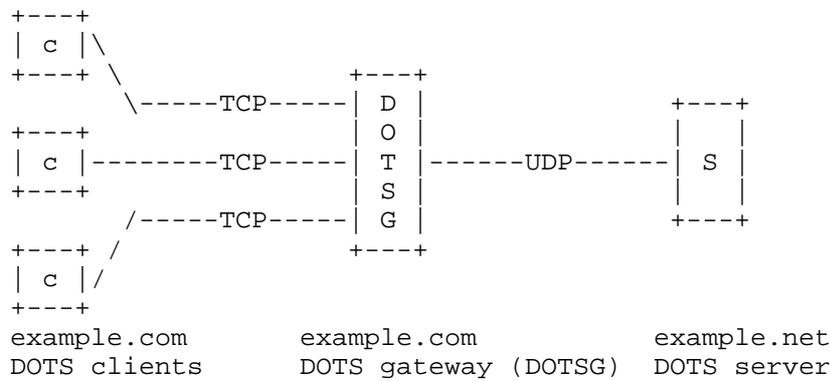   the gateway and its upstream agents is opaque to the initial clients.

```
        +---+
        | c |\
        +---+ \              +---+
              \-----TCP-----| D |                    +---+
        +---+              | O |                    |   |
        | c |--------TCP-----| T |------UDP------| S |
        +---+              | S |                    |   |
              /-----TCP-----| G |                    +---+
        +---+ /              +---+
        | c |/
        +---+
        example.com       example.com         example.net
        DOTS clients      DOTS gateway (DOTSG) DOTS server
```

              Figure 7: Client-Side Gateway with Aggregation

```
        +---+
        | c |\
        +---+ \              +---+
              \-----TCP-----| D |------UDP------+---+
        +---+              | O |                    |   |
        | c |--------TCP-----| T |------UDP------| S |
        +---+              | S |                    |   |
              /-----TCP-----| G |------UDP------+---+
        +---+ /              +---+
        | c |/
        +---+
        example.com       example.com         example.net
        DOTS clients      DOTS gateway (DOTSG) DOTS server
```
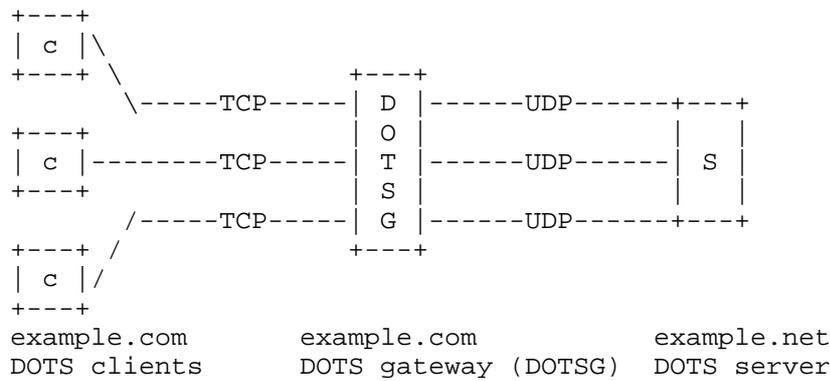
             Figure 8: Client-Side Gateway without Aggregation

This may similarly be deployed in the inverse scenario where the
gateway resides in the server-side domain and may be used to
terminate and/or aggregate multiple clients to single transport as
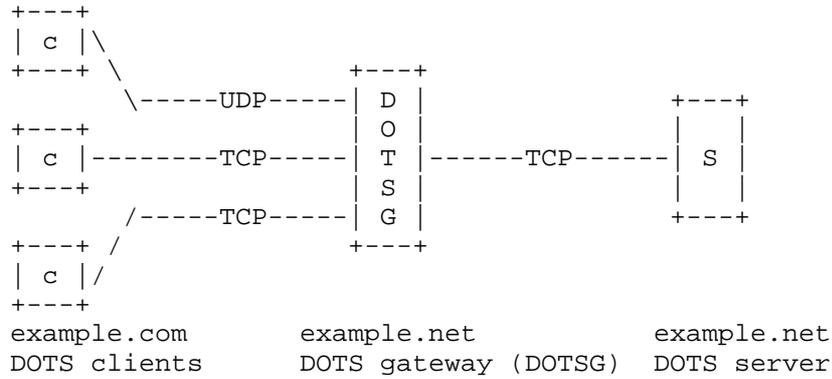shown in figures Figure 9 and Figure 10.

```
        +---+
        | c |\
        +---+ \            +---+
               \-----UDP-----| D |               +---+
        +---+              | O |                 |   |
        | c |--------TCP-----| T |------TCP------| S |
        +---+              | S |                 |   |
              /-----TCP-----| G |               +---+
        +---+ /             +---+
        | c |/
        +---+
        example.com        example.net        example.net
        DOTS clients       DOTS gateway (DOTSG) DOTS server
```

              Figure 9: Server-Side Gateway with Aggregation

```
        +---+
        | c |\
        +---+ \            +---+
               \-----UDP-----| D |------TCP------+---+
        +---+              | O |                 |   |
        | c |--------TCP-----| T |------TCP------| S |
        +---+              | S |                 |   |
              /-----UDP-----| G |------TCP------+---+
        +---+ /             +---+
        | c |/
        +---+
        example.com        example.net        example.net
        DOTS clients       DOTS gateway (DOTSG) DOTS server
```
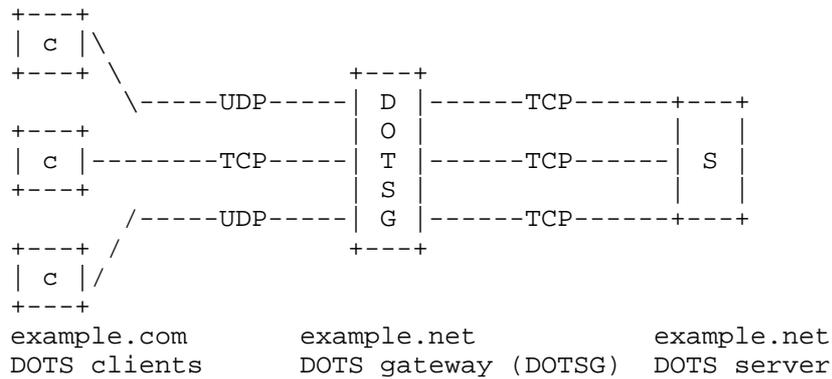
              Figure 10: Server-Side Gateway without Aggregation

   This document anticipates scenarios involving multiple DOTS gateways.
   An example is a DOTS gateway at the network client's side, and
   another one at the server side. The first gateway can be located at
   a CPE to aggregate requests from multiple DOTS clients enabled in an
   enterprise network. The second DOTS gateway is deployed on the
   provider side. This scenario can be seen as a combination of the
   client-side and server-side scenarios.

3. Concepts

3.1. DOTS Sessions

   In order for DOTS to be effective as a vehicle for DDoS mitigation
   requests, one or more DOTS clients must establish ongoing
   communication with one or more DOTS servers. While the preconditions
   for enabling DOTS in or among network domains may also involve
   business relationships, service level agreements, or other formal or
   informal understandings between network operators, such
   considerations are out of scope for this document.

   A DOTS session is established to support bilateral exchange of data
   between an associated DOTS client and a DOTS server. In the DOTS
   architecture, data is exchanged between DOTS agents over signal and
   data channels. Regardless, a DOTS session is characterized by the
   presence of an established signal channel. A data channel may be
   established as well, however it is not a prerequisite. Conversely, a
   DOTS session cannot exist without an established signal channel: when
   an established signal channel is terminated for any reason, the DOTS
   session is also said to be terminated.

3.1.1. Preconditions

   Prior to establishing a DOTS session between agents, the owners of
   the networks, domains, services or applications involved are assumed
   to have agreed upon the terms of the relationship involved. Such
   agreements are out of scope for this document, but must be in place
   for a functional DOTS architecture.

   It is assumed that as part of any DOTS service agreement, the DOTS
   client is provided with all data and metadata required to establish
   communication with the DOTS server. Such data and metadata would
   include any cryptographic information necessary to meet the message
   confidentiality, integrity and authenticity requirement in
   [I-D.ietf-dots-requirements], and might also include the pool of DOTS
   server addresses and ports the DOTS client should use for signal and
   data channel messaging.

3.1.2. Establishing the DOTS Session

   With the required business agreements in place, the DOTS client
   initiates a signal session by contacting its DOTS server(s) over the
   signal channel and (possibly) the data channel. To allow for DOTS
   service flexibility, neither the order of contact nor the time
   interval between channel creations is specified. A DOTS client MAY
   establish signal channel first, and then data channel, or vice versa.

The methods by which a DOTS client receives the address and
associated service details of the DOTS server are not prescribed by
this document. For example, a DOTS client may be directly configured
to use a specific DOTS server IP address and port, and directly
provided with any data necessary to satisfy the Peer Mutual
Authentication requirement in [I-D.ietf-dots-requirements], such as
symmetric or asymmetric keys, usernames and passwords, etc. All
configuration and authentication information in this scenario is
provided out-of-band by the domain operating the DOTS server.

At the other extreme, the architecture in this document allows for a
form of DOTS client auto-provisioning. For example, the domain
operating the DOTS server or servers might provide the client domain
only with symmetric or asymmetric keys to authenticate the
provisioned DOTS clients. Only the keys would then be directly
configured on DOTS clients, but the remaining configuration required
to provision the DOTS clients could be learned through mechanisms
similar to DNS SRV [RFC2782] or DNS Service Discovery [RFC6763].

The DOTS client SHOULD successfully authenticate and exchange
messages with the DOTS server over both signal and (if used) data
channel as soon as possible to confirm that both channels are
operational.

As described in [I-D.ietf-dots-requirements], the DOTS client can
configure preferred values for acceptable signal loss, mitigation
lifetime, and heartbeat intervals when establishing the DOTS session.
A DOTS session is not active until DOTS agents have agreed on the
values for these DOTS session parameters, a process defined by the
protocol.

Once the DOTS client begins receiving DOTS server signals, the DOTS
session is active. At any time during the DOTS session, the DOTS
client may use the data channel to manage aliases, manage black- and
white-listed prefixes or addresses, leverage vendor-specific
extensions, and so on. Note that unlike the signal channel, there is
no requirement that the data channel remains operational in attack
conditions (See Data Channel Requirements,
[I-D.ietf-dots-requirements]).

3.1.3. Maintaining the DOTS Session

DOTS clients and servers periodically send heartbeats to each other
over the signal channel, per Operational Requirements discussed in
[I-D.ietf-dots-requirements]. DOTS agent operators SHOULD configure
the heartbeat interval such that the frequency does not lead to
accidental denials of service due to the overwhelming number of
heartbeats a DOTS agent must field.

Either DOTS agent may consider a DOTS session terminated in the
extended absence of a heartbeat from its peer agent. The period of
that absence will be established in the protocol definition.

3.2. Modes of Signaling

This section examines the modes of signaling between agents in a DOTS
architecture.

3.2.1. Direct Signaling

A DOTS session may take the form of direct signaling between the DOTS
clients and servers, as shown in Figure 11.

```
      +-------------+                         +-------------+
      | DOTS client |<------signal session------>| DOTS server |
      +-------------+                         +-------------+
```
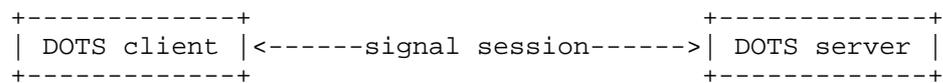
Figure 11: Direct Signaling

In a direct DOTS session, the DOTS client and server are
communicating directly. Direct signaling may exist inter- or intra-
domain. The DOTS session is abstracted from the underlying networks
or network elements the signals traverse: in direct signaling, the
DOTS client and server are logically adjacent.

3.2.2. Redirected Signaling

In certain circumstances, a DOTS server may want to redirect a DOTS
client to an alternative DOTS server for a DOTS session. Such
circumstances include but are not limited to:

o Maximum number of DOTS sessions with clients has been reached;

o Mitigation capacity exhaustion in the mitigator with which the
   specific DOTS server is communicating;

o Mitigator outage or other downtime, such as scheduled maintenance;

o Scheduled DOTS server maintenance;

o Scheduled modifications to the network path between DOTS server
   and DOTS client.

A basic redirected DOTS session resembles the following, as shown in
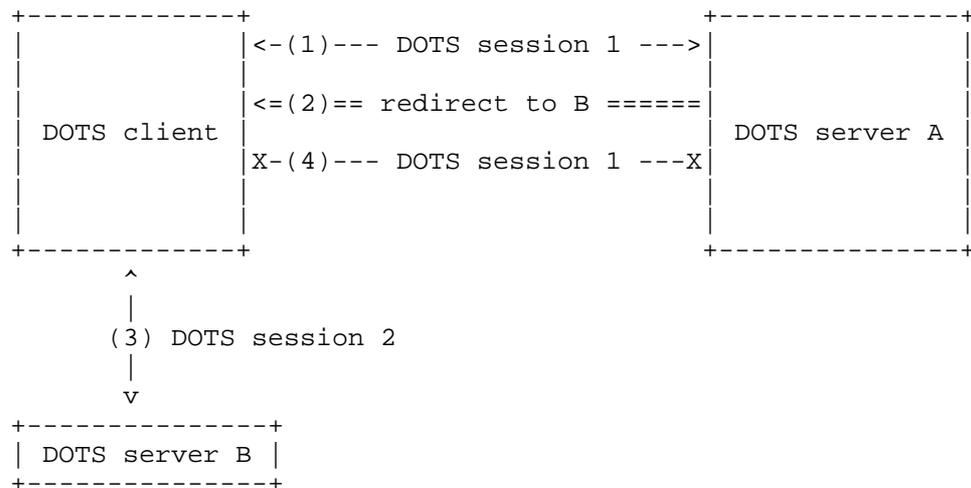Figure 12.

```
     +------------+                              +--------------+
     |            |<-(1)--- DOTS session 1 --->|              |
     |            |        |                     |              |
     |            |<=(2)== redirect to B ======|              |
     | DOTS client|        |                     | DOTS server A|
     |            |X-(4)--- DOTS session 1 ---X|              |
     |            |        |                     |              |
     |            |        |                     |              |
     +------------+                              +--------------+
            ^
            |
         (3) DOTS session 2
            |
            v
     +---------------+
     | DOTS server B |
     +---------------+
```

                    Figure 12: Redirected Signaling

   1. Previously established DOTS session 1 exists between a DOTS
       client and DOTS server A.

   2. DOTS server A sends a server signal redirecting the client to
       DOTS server B.

   3. If the DOTS client does not already have a separate DOTS session
       with the redirection target, the DOTS client initiates and
       establishes DOTS session 2 with DOTS server B.

   4. Having redirected the DOTS client, DOTS server A ceases sending
       server signals. The DOTS client likewise stops sending client
       signals to DOTS server A. DOTS session 1 is terminated.

3.2.3. Recursive Signaling

   DOTS is centered around improving the speed and efficiency of
   coordinated response to DDoS attacks. One scenario not yet discussed
   involves coordination among federated domains operating DOTS servers
   and mitigators.

   In the course of normal DOTS operations, a DOTS client communicates
   the need for mitigation to a DOTS server, and that server initiates
   mitigation on a mitigator with which the server has an established
   service relationship. The operator of the mitigator may in turn
   monitor mitigation performance and capacity, as the attack being
   mitigated may grow in severity beyond the mitigating domain's
   capabilities.

The operator of the mitigator has limited options in the event a DOTS
client-requested mitigation is being overwhelmed by the severity of
the attack. Out-of-scope business or service level agreements may
permit the mitigating domain to drop the mitigation and let attack
traffic flow unchecked to the target, but this only encourages attack
escalation. In the case where the mitigating domain is the upstream
service provider for the attack target, this may mean the mitigating
domain and its other services and users continue to suffer the
incidental effects of the attack.

A recursive signaling model as shown in Figure 13 offers an
alternative. In a variation of the use case "End-customer with a
single upstream transit provider offering DDoS mitigation services"
described in [I-D.ietf-dots-use-cases], a domain operating a DOTS
server and mitigator also operates a DOTS client. This DOTS client
has an established DOTS session with a DOTS server belonging to a
separate administrative domain.

With these preconditions in place, the operator of the mitigator
being overwhelmed or otherwise performing inadequately may request
mitigation for the attack target from this separate DOTS-aware
domain. Such a request recurses the originating mitigation request
to the secondary DOTS server, in the hope of building a cumulative
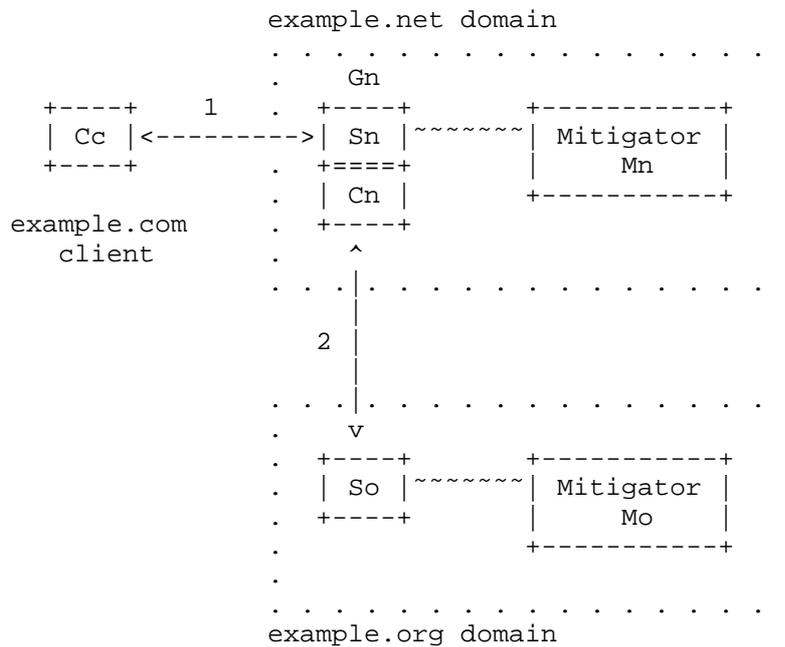mitigation against the attack.

```
                        example.net domain
                    . . . . . . . . . . . . . . . .
                    .        Gn                     .
         +----+    1 .  +----+       +-----------+   .
         | Cc |<---------->| Sn |~~~~~~~| Mitigator |   .
         +----+    .  +====+       |     Mn    |   .
                    .  | Cn |       +-----------+   .
         example.com .  +----+                      .
            client   .    ^                          .
                    . . .|. . . . . . . . . . . . . .
                    .     |                          .
                          |
                    2 |
                          |
                    . . .|. . . . . . . . . . . . . .
                    .     v                          .
                    .  +----+       +-----------+   .
                    .  | So |~~~~~~~| Mitigator |   .
                    .  +----+       |     Mo    |   .
                    .                +-----------+   .
                    .                                .
                    .                                .
                    . . . . . . . . . . . . . . . .
                        example.org domain
```

                     Figure 13: Recursive Signaling

   In Figure 13, client Cc signals a request for mitigation across
   inter-domain DOTS session 1 to the DOTS server Sn belonging to the
   example.net domain. DOTS server Sn enables mitigation on mitigator
   Mn. DOTS server Sn is half of DOTS gateway Gn, being deployed
   logically back-to-back with DOTS client Cn, which has pre-existing
   inter-domain DOTS session 2 with the DOTS server So belonging to the
   example.org domain. At any point, DOTS server Sn MAY recurse an on-
   going mitigation request through DOTS client Cn to DOTS server So, in
   the expectation that mitigator Mo will be activated to aid in the
   defense of the attack target.

   Recursive signaling is opaque to the DOTS client. To maximize
   mitigation visibility to the DOTS client, however, the recursing
   domain SHOULD provide recursed mitigation feedback in signals
   reporting on mitigation status to the DOTS client. For example, the
   recursing domain's mitigator should incorporate into mitigation
   status messages available metrics such as dropped packet or byte
   counts from the recursed mitigation.

   DOTS clients involved in recursive signaling must be able to withdraw
   requests for mitigation without warning or justification, per
   [I-D.ietf-dots-requirements].

   Operators recursing mitigation requests MAY maintain the recursed
   mitigation for a brief, protocol-defined period in the event the DOTS
   client originating the mitigation withdraws its request for help, as
   per the discussion of managing mitigation toggling in the operational
   requirements ([I-D.ietf-dots-requirements]).

   Deployment of recursive signaling may result in traffic redirection,
   examination and mitigation extending beyond the initial bilateral
   relationship between DOTS client and DOTS server. As such, client
   control over the network path of mitigated traffic may be reduced.
   DOTS client operators should be aware of any privacy concerns, and
   work with DOTS server operators employing recursive signaling to
   ensure shared sensitive material is suitably protected.

3.2.4. Anycast Signaling

   The DOTS architecture does not assume the availability of anycast
   within a DOTS deployment, but neither does the architecture exclude
   it. Domains operating DOTS servers MAY deploy DOTS servers with an
   anycast Service Address as described in BCP 126 [RFC4786]. In such a
   deployment, DOTS clients connecting to the DOTS Service Address may
   be communicating with distinct DOTS servers, depending on the network
   configuration at the time the DOTS clients connect. Among other
   benefits, anycast signaling potentially offers the following:

   o Simplified DOTS client configuration, including service discovery
      through the methods described in [RFC7094]. In this scenario, the
      "instance discovery" message would be a DOTS client initiating a
      DOTS session to the DOTS server anycast Service Address, to which
      the DOTS server would reply with a redirection to the DOTS server
      unicast address the client should use for DOTS.

   o Region- or customer-specific deployments, in which the DOTS
      Service Addresses route to distinct DOTS servers depending on the
      client region or the customer network in which a DOTS client
      resides.

   o Operational resiliency, spreading DOTS signaling traffic across
      the DOTS server domain's networks, and thereby also reducing the
      potential attack surface, as described in BCP 126 [RFC4786].

3.2.4.1. Anycast Signaling Considerations

   As long as network configuration remains stable, anycast DOTS
   signaling is to the individual DOTS client indistinct from direct
   signaling. However, the operational challenges inherent in anycast
   signaling are anything but negligible, and DOTS server operators must
   carefully weigh the risks against the benefits before deploying.

While the DOTS signal channel primarily operates over UDP per
[I-D.ietf-dots-requirements], the signal channel also requires mutual
authentication between DOTS agents, with associated security state on
both ends.

Network instability is of particular concern with anycast signaling,
as DOTS signal channels are expected to be long-lived, and
potentially operating under congested network conditions caused by a
volumetric DDoS attack.

For example, a network configuration altering the route to the DOTS
server during active anycast signaling may cause the DOTS client to
send messages to a DOTS server other than the one with which it
initially established a signaling session. That second DOTS server
may not have the security state of the existing session, forcing the
DOTS client to initialize a new DOTS session. This challenge might
in part be mitigated by use of pre-shared keys and session resumption
[RFC5246][RFC6347], but keying material must be available to all DOTS
servers sharing the anycast Service Address in that case.

While the DOTS client will try to establish a new DOTS session with
the DOTS server now acting as the anycast DOTS Service Address, the
link between DOTS client and server may be congested with attack
traffic, making signal session establishment difficult. In such a
scenario, anycast Service Address instability becomes a sort of
signal session flapping, with obvious negative consequences for the
DOTS deployment.

Anycast signaling deployments similarly must also take into account
active mitigations. Active mitigations initiated through a DOTS
session may involve diverting traffic to a scrubbing center. If the
DOTS session flaps due to anycast changes as described above,
mitigation may also flap as the DOTS servers sharing the anycast DOTS
service address toggles mitigation on detecting DOTS session loss,
depending on whether the client has configured mitigation on loss of
signal.

3.2.5. Signaling Considerations for Network Address Translation

Network address translators (NATs) are expected to be a common
feature of DOTS deployments. The Middlebox Traversal Guidelines in
[RFC8085] include general NAT considerations for DOTS deployyements
when the signal channel is established over UDP.

Additional DOTS-specific considerations arise when NATs are part of
the DOTS architecture. For example, DDoS attack detection behind a
NAT will detect attacks against internal addresses. A DOTS client
subsequently asked to request mitigation for the attacked scope of

addresses cannot reasonably perform the task, due to the lack of
externally routable addresses in the mitigation scope.

The following considerations do not cover all possible scenarios, but
are meant rather to highlight anticipated common issues when
signaling through NATs.

3.2.5.1. Direct Provisioning of Internal-to-External Address Mappings

Operators may circumvent the problem of translating internal
addresses or prefixes to externally routable mitigation scopes by
directly provisioning the mappings of external addresses to internal
protected resources on the DOTS client. When the operator requests
mitigation scoped for internal addresses, directly or through
automated means, the DOTS client looks up the matching external
addresses or prefixes, and issues a mitigation request scoped to that
externally routable information.

When directly provisioning the address mappings, operators must
ensure the mappings remain up to date, or risk losing the ability to
request accurate mitigation scopes. This document does not prescribe
the method by which mappings are maintained once they are provisioned
on the DOTS client.

3.2.5.2. Resolving Public Mitigation Scope with Session Traversal
Utilities (STUN)

Internal resources may discover their external address through a STUN
Binding request/response transaction, as described in [RFC5389].
After learning its reflexive transport address from the STUN server,
the internal service can export its external/internal address pair to
the DOTS client, allowing the DOTS client to request mitigation with
the correct external scope, as depicted in Figure 14. The mechanism
for provisioning the DOTS client with the address pair is
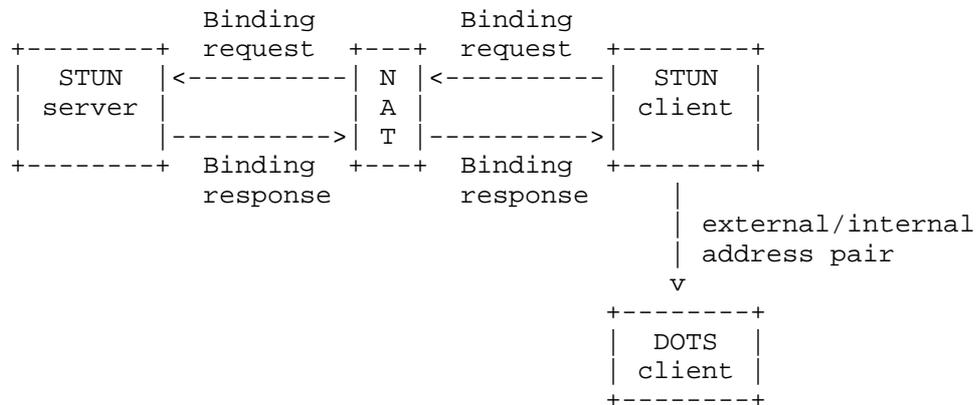unspecified.

```
                   Binding         Binding
        +--------+ request +---+ request +--------+
        | STUN   |<--------| N |<--------| STUN   |
        | server |         | A |         | client |
        |        |-------->| T |-------->|        |
        +--------+ Binding +---+ Binding +--------+
                   response       response   |
                                             | external/internal
                                             | address pair
                                             v
                                       +--------+
                                       | DOTS   |
                                       | client |
                                       +--------+
```

               Figure 14: Resolving mitigation scope with STUN

3.2.5.3. Resolving Requested Mitigation Scope with DNS

   DOTS supports mitigation scoped to DNS names. As discussed in
   [RFC3235], using DNS names instead of IP addresses potentially avoids
   the address translation problem, as long as the name is internally
   and externally resolvable by the same name. For example, a detected
   attack's internal target address can be mapped to a DNS name through
   a reverse lookup. The DNS name returned by the reverse lookup can
   then be provided to the DOTS client as the external scope for
   mitigation.

3.3. Triggering Requests for Mitigation

   [I-D.ietf-dots-requirements] places no limitation on the
   circumstances in which a DOTS client operator may request mitigation,
   nor does it demand justification for any mitigation request, thereby
   reserving operational control over DDoS defense for the domain
   requesting mitigation. This architecture likewise does not prescribe
   the network conditions and mechanisms triggering a mitigation request
   from a DOTS client.

   However, considering selected possible mitigation triggers from an
   architectural perspective offers a model for alternative or
   unanticipated triggers for DOTS deployments. In all cases, what
   network conditions merit a mitigation request are at the discretion
   of the DOTS client operator.

   The mitigation request itself is defined by DOTS, however the
   interfaces required to trigger the mitigation request in the
   following scenarios are implementation-specific.

3.3.1. Manual Mitigation Request

   A DOTS client operator may manually prepare a request for mitigation,
   including scope and duration, and manually instruct the DOTS client
   to send the mitigation request to the DOTS server. In context, a
   manual request is a request directly issued by the operator without
   automated decision-making performed by a device interacting with the
   DOTS client. Modes of manual mitigation requests include an operator
   entering a command into a text interface, or directly interacting
   with a graphical interface to send the request.

   An operator might do this, for example, in response to notice of an
   attack delivered by attack detection equipment or software, and the
   alerting detector lacks interfaces or is not configured to use
   available interfaces to translate the alert to a mitigation request
   automatically.

   In a variation of the above scenario, the operator may have
   preconfigured on the DOTS client mitigation requests for various
   resources in the operator's domain. When notified of an attack, the
   DOTS client operator manually instructs the DOTS client to send the
   relevant preconfigured mitigation request for the resources under
   attack.

   A further variant involves recursive signaling, as described in
   Section 3.2.3. The DOTS client in this case is the second half of a
   DOTS gateway (back-to-back DOTS server and client). As in the
   previous scenario, the scope and duration of the mitigation request
   are pre-existing, but in this case are derived from the mitigation
   request received from a downstream DOTS client by the DOTS server.
   Assuming the preconditions required by Section 3.2.3 are in place,
   the DOTS gateway operator may at any time manually request mitigation
   from an upstream DOTS server, sending a mitigation request derived
   from the downstream DOTS client's request.

   The motivations for a DOTS client operator to request mitigation
   manually are not prescribed by this architecture, but are expected to
   include some of the following:

   o Notice of an attack delivered via e-mail or alternative messaging

   o Notice of an attack delivered via phone call

   o Notice of an attack delivered through the interface(s) of
      networking monitoring software deployed in the operator's domain

   o Manual monitoring of network behavior through network monitoring
      software

3.3.2. Automated Conditional Mitigation Request

   Unlike manual mitigation requests, which depend entirely on the DOTS
   client operator's capacity to react with speed and accuracy to every
   detected or detectable attack, mitigation requests triggered by
   detected attack conditions reduce the operational burden on the DOTS
   client operator, and minimize the latency between attack detection
   and the start of mitigation.

   Mitigation requests are triggered in this scenario by operator-
   specified network conditions. Attack detection is deployment-
   specific, and not constrained by this architecture. Similarly the
   specifics of a condition are left to the discretion of the operator,
   though common conditions meriting mitigation include the following:

   o Detected attack exceeding a rate in packets per second (pps).

   o Detected attack exceeding a rate in bytes per second (bps).

   o Detected resource exhaustion in an attack target.

   o Detected resource exhaustion in the local domain's mitigator.

   o Number of open connections to an attack target.

   o Number of attack sources in a given attack.

   o Number of active attacks against targets in the operator's domain.

   o Conditional detection developed through arbitrary statistical
      analysis or deep learning techniques.

   o Any combination of the above.

   When automated conditional mitigation requests are enabled,
   violations of any of the above conditions, or any additional
   operator-defined conditions, will trigger a mitigation request from
   the DOTS client to the DOTS server. The interfaces between the
   application detecting the condition violation and the DOTS client are
   implementation-specific.

3.3.3. Automated Mitigation on Loss of Signal

   To maintain a DOTS session, the DOTS client and the DOTS server
   exchange regular but infrequent messages across the signal channel.
   In the absence of an attack, the probability of message loss in the
   signaling channel should be extremely low. Under attack conditions,

however, some signal loss may be anticipated as attack traffic
congests the link, depending on the attack type.

While [I-D.ietf-dots-requirements] specifies the DOTS protocol be
robust when signaling under attack conditions, there are nevertheless
scenarios in which the DOTS signal is lost in spite of protocol best
efforts. To handle such scenarios, a DOTS operator may configure the
DOTS session to trigger mitigation when the DOTS server ceases
receiving DOTS client signals (or vice versa) beyond the miss count
or period permitted by the protocol.

The impact of mitigating due to loss of signal in either direction
must be considered carefully before enabling it. Signal loss is not
caused by links congested with attack traffic alone, and as such
mitigation requests triggered by signal channel degradation in either
direction may incur unnecessary costs, in network performance and
operational expense alike.

4. Security Considerations

This section describes identified security considerations for the
DOTS architecture.

DOTS is at risk from three primary attack vectors: agent
impersonation, traffic injection and signal blocking. These vectors
may be exploited individually or in concert by an attacker to
confuse, disable, take information from, or otherwise inhibit DOTS
agents.

Any attacker with the ability to impersonate a legitimate DOTS client
or server or, indeed, inject false messages into the stream may
potentially trigger/withdraw traffic redirection, trigger/cancel
mitigation activities or subvert black/whitelists. From an
architectural standpoint, operators SHOULD ensure best current
practices for secure communication are observed for data and signal
channel confidentiality, integrity and authenticity. Care must be
taken to ensure transmission is protected by appropriately secure
means, reducing attack surface by exposing only the minimal required
services or interfaces. Similarly, received data at rest SHOULD be
stored with a satisfactory degree of security.

As many mitigation systems employ diversion to scrub attack traffic,
operators of DOTS agents SHOULD ensure DOTS sessions are resistant to
Man-in-the-Middle (MitM) attacks. An attacker with control of a DOTS
client may negatively influence network traffic by requesting and
withdrawing requests for mitigation for particular prefixes, leading
to route or DNS flapping.

   Any attack targeting the availability of DOTS servers may disrupt the
   ability of the system to receive and process DOTS signals resulting
   in failure to fulfill a mitigation request. DOTS agents SHOULD be
   given adequate protections, again in accordance with best current
   practices for network and host security.

5. Contributors

   Mohamed Boucadair
      Orange

      mohamed.boucadair@orange.com

6. Acknowledgments

   Thanks to Matt Richardson and Med Boucadair for their comments and
   suggestions.

7. References

7.1. Normative References

   [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
              Requirement Levels", BCP 14, RFC 2119,
              DOI 10.17487/RFC2119, March 1997,
              <https://www.rfc-editor.org/info/rfc2119>.

7.2. Informative References

   [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-14 (work in
              progress), February 2018.

   [I-D.ietf-dots-use-cases]
              Dobbins, R., Migault, D., Fouant, S., Moskowitz, R.,
              Teague, N., Xia, L., and K. Nishizuka, "Use cases for DDoS
              Open Threat Signaling", draft-ietf-dots-use-cases-09 (work
              in progress), November 2017.

   [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768,
              DOI 10.17487/RFC0768, August 1980,
              <https://www.rfc-editor.org/info/rfc768>.

   [RFC0793] Postel, J., "Transmission Control Protocol", STD 7,
              RFC 793, DOI 10.17487/RFC0793, September 1981,
              <https://www.rfc-editor.org/info/rfc793>.

   [RFC1035] Mockapetris, P., "Domain names - implementation and
              specification", STD 13, RFC 1035, DOI 10.17487/RFC1035,
              November 1987, <https://www.rfc-editor.org/info/rfc1035>.

   [RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for
              specifying the location of services (DNS SRV)", RFC 2782,
              DOI 10.17487/RFC2782, February 2000,
              <https://www.rfc-editor.org/info/rfc2782>.

   [RFC3235] Senie, D., "Network Address Translator (NAT)-Friendly
              Application Design Guidelines", RFC 3235,
              DOI 10.17487/RFC3235, January 2002,
              <https://www.rfc-editor.org/info/rfc3235>.

   [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston,
              A., Peterson, J., Sparks, R., Handley, M., and E.
              Schooler, "SIP: Session Initiation Protocol", RFC 3261,
              DOI 10.17487/RFC3261, June 2002,
              <https://www.rfc-editor.org/info/rfc3261>.

   [RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A
              Border Gateway Protocol 4 (BGP-4)", RFC 4271,
              DOI 10.17487/RFC4271, January 2006,
              <https://www.rfc-editor.org/info/rfc4271>.

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

   [RFC4786] Abley, J. and K. Lindqvist, "Operation of Anycast
              Services", BCP 126, RFC 4786, DOI 10.17487/RFC4786,
              December 2006, <https://www.rfc-editor.org/info/rfc4786>.

   [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security
              (TLS) Protocol Version 1.2", RFC 5246,
              DOI 10.17487/RFC5246, August 2008,
              <https://www.rfc-editor.org/info/rfc5246>.

   [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing,
              "Session Traversal Utilities for NAT (STUN)", RFC 5389,
              DOI 10.17487/RFC5389, October 2008,
              <https://www.rfc-editor.org/info/rfc5389>.

   [RFC6347] Rescorla, E. and N. Modadugu, "Datagram Transport Layer
              Security Version 1.2", RFC 6347, DOI 10.17487/RFC6347,
              January 2012, <https://www.rfc-editor.org/info/rfc6347>.

   [RFC6763] Cheshire, S. and M. Krochmal, "DNS-Based Service
              Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013,
              <https://www.rfc-editor.org/info/rfc6763>.

   [RFC7092] Kaplan, H. and V. Pascual, "A Taxonomy of Session
              Initiation Protocol (SIP) Back-to-Back User Agents",
              RFC 7092, DOI 10.17487/RFC7092, December 2013,
              <https://www.rfc-editor.org/info/rfc7092>.

   [RFC7094] McPherson, D., Oran, D., Thaler, D., and E. Osterweil,
              "Architectural Considerations of IP Anycast", RFC 7094,
              DOI 10.17487/RFC7094, January 2014,
              <https://www.rfc-editor.org/info/rfc7094>.

   [RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage
              Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085,
              March 2017, <https://www.rfc-editor.org/info/rfc8085>.

Authors' Addresses

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

   EMail: amortensen@arbor.net

   Flemming Andreasen
   Cisco
   United States

   EMail: fandreas@cisco.com

   Tirumaleswar Reddy
   McAfee, Inc.
   Embassy Golf Link Business Park
   Bangalore, Karnataka 560071
   India

   EMail: tireddy@cisco.com

   Christopher Gray
   United States

   EMail: Christopher_Gray3@cable.comcast.com

   Rich Compton
   Charter

   EMail: Rich.Compton@charter.com

   Nik Teague
   Verisign
   United States

   EMail: nteague@verisign.com

DOTS   A. Mortensen
Internet-Draft   Arbor Networks, Inc.
Intended status: Informational   R. Moskowitz
Expires: September 14, 2017   HTT Consulting
                                                               T. Reddy
                                                     Cisco Systems, Inc.
                                                         March 13, 2017

         Distributed Denial of Service (DDoS) Open Threat Signaling Requirements
                       draft-ietf-dots-requirements-04

   Abstract

      This document defines the requirements for the Distributed Denial of
      Service (DDoS) Open Threat Signaling (DOTS) protocols coordinating
      attack response against DDoS attacks.

   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 14, 2017.

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

Table of Contents

1. Introduction

1.1. Context and Motivation

   Distributed Denial of Service (DDoS) attacks continue to plague
   networks around the globe, from Tier-1 service providers on down to
   enterprises and small businesses. Attack scale and frequency
   similarly have continued to increase, in part as a result of software
   vulnerabilities leading to reflection and amplification attacks.
   Once-staggering attack traffic volume is now the norm, and the impact
   of larger-scale attacks attract the attention of international press
   agencies.

   The greater impact of contemporary DDoS attacks has led to increased
   focus on coordinated attack response. Many institutions and
   enterprises lack the resources or expertise to operate on-premise

attack mitigation solutions themselves, or simply find themselves
constrained by local bandwidth limitations. To address such gaps,
security service providers have begun to offer on-demand traffic
scrubbing services, which aim to separate the DDoS traffic from
legitimate traffic and forward only the latter. Today each such
service offers its own interface for subscribers to request attack
mitigation, tying subscribers to proprietary implementations while
also limiting the subset of network elements capable of participating
in the attack response. As a result of incompatibility across
services, attack responses may be fragmentary or otherwise
incomplete, leaving key players in the attack path unable to assist
in the defense.

The lack of a common method to coordinate a real-time response among
involved actors and network domains inhibits the speed and
effectiveness of DDoS attack mitigation. This document describes the
required characteristics of a DOTS protocol enabling requests for
DDoS attack mitigation, reducing attack impact and leading to more
efficient defensive strategies.

DOTS communicates the need for defensive action in anticipation of or
in response to an attack, but does not dictate the form any defensive
action takes. DOTS supplements calls for help with pertinent details
about the detected attack, allowing entities participating in DOTS to
form ad hoc, adaptive alliances against DDoS attacks as described in
the DOTS use cases [I-D.ietf-dots-use-cases]. The requirements in
this document are derived from those use cases and
[I-D.ietf-dots-architecture].

1.2. 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].

This document adopts the following terms:

DDoS: A distributed denial-of-service attack, in which traffic
   originating from multiple sources are directed at a target on a
   network. DDoS attacks are intended to cause a negative impact on
   the availability of servers, services, applications, and/or other
   functionality of an attack target. Denial-of-service
   considerations are discussed in detail in [RFC4732].

DDoS attack target: A network connected entity with a finite set of
   resources, such as network bandwidth, memory or CPU, that is the
   focus of a DDoS attack. Potential targets include network
   elements, network links, servers, and services.

DDoS attack telemetry: Collected measurements and behavioral
   characteristics defining the nature of a DDoS attack.

Countermeasure: An action or set of actions taken to recognize and
   filter out DDoS attack traffic while passing legitimate traffic to
   the attack target.

Mitigation: A set of countermeasures enforced against traffic
   destined for the target or targets of a detected or reported DDoS
   attack, where countermeasure enforcement is managed by an entity
   in the network path between attack sources and the attack target.
   Mitigation methodology is out of scope for this document.

Mitigator: An entity, typically a network element, capable of
   performing mitigation of a detected or reported DDoS attack. For
   the purposes of this document, this entity is a black box capable
   of mitigation, making no assumptions about availability or design
   of countermeasures, nor about the programmable interface between
   this entity and other network elements. The mitigator and DOTS
   server are assumed to belong to the same administrative entity.

DOTS client: A DOTS-aware software module responsible for requesting
   attack response coordination with other DOTS-aware elements.

DOTS server: A DOTS-aware software module handling and responding to
   messages from DOTS clients. The DOTS server SHOULD enable
   mitigation on behalf of the DOTS client, if requested, by
   communicating the DOTS client's request to the mitigator and
   returning selected mitigator feedback to the requesting DOTS
   client. A DOTS server MAY also be a mitigator.

DOTS agent: Any DOTS-aware software module capable of participating
   in a DOTS signaling session.

DOTS gateway: A logical DOTS agent resulting from the logical
   concatenation of a DOTS server and a DOTS client, analogous to a
   SIP Back-to-Back User Agent (B2BUA) [RFC3261]. DOTS gateways are
   discussed in detail in [I-D.ietf-dots-architecture].

Signal channel: A bidirectional, mutually authenticated
   communication channel between DOTS agents characterized by
   resilience even in conditions leading to severe packet loss, such
   as a volumetric DDoS attack causing network congestion.

DOTS signal: A concise authenticated status/control message
   transmitted between DOTS agents, used to indicate client's need
   for mitigation, as well as to convey the status of any requested
   mitigation.

Heartbeat: A message transmitted between DOTS agents over the signal
   channel, used as a keep-alive and to measure peer health.

Client signal: A message sent from a DOTS client to a DOTS server
   over the signal channel, indicating the DOTS client's need for
   mitigation, as well as the scope of any requested mitigation,
   optionally including additional attack details to supplement
   server-initiated mitigation.

Server signal: A message sent from a DOTS server to a DOTS client
   over the signal channel. Note that a server signal is not a
   response to client signal, but a DOTS server-initiated status
   message sent to DOTS clients with which the server has established
   signaling sessions.

Data channel: A secure communication layer between DOTS clients and
   DOTS servers used for infrequent bulk exchange of data not easily
   or appropriately communicated through the signal channel under
   attack conditions.

Filter: A policy matching a network traffic flow or set of flows and
   rate-limiting or discarding matching traffic.

Blacklist: A filter list of addresses, prefixes and/or other
   identifiers indicating sources from which traffic should be
   blocked, regardless of traffic content.

Whitelist: A list of addresses, prefixes and/or other identifiers
   from indicating sources from which traffic should always be
   allowed, regardless of contradictory data gleaned in a detected
   attack.

Multi-homed DOTS client: A DOTS client exchanging messages with
   multiple DOTS servers, each in a separate administrative domain.

2. Requirements

   This section describes the required features and characteristics of
   the DOTS protocol.

   DOTS is an advisory protocol. An active DDoS attack against the
   entity controlling the DOTS client need not be present before
   establishing DOTS communication between DOTS agents. Indeed,
   establishing a relationship with peer DOTS agents during normal
   network conditions provides the foundation for more rapid attack
   response against future attacks, as all interactions setting up DOTS,
   including any business or service level agreements, are already
   complete.

DOTS must at a minimum make it possible for a DOTS client to request
a DOTS server's aid in mounting a coordinated defense against a
suspected attack, signaling within or between domains as requested by
local operators. DOTS clients should similarly be able to withdraw
aid requests. DOTS requires no justification from DOTS clients for
requests for help, nor do DOTS clients need to justify withdrawing
help requests: the decision is local to the DOTS clients' domain.
Regular feedback between DOTS clients and DOTS server supplement the
defensive alliance by maintaining a common understanding of DOTS peer
health and activity. Bidirectional communication between DOTS
clients and DOTS servers is therefore critical.

Yet DOTS must also work with a set of competing operational goals.
On the one hand, the protocol must be resilient under extremely
hostile network conditions, providing continued contact between DOTS
agents even as attack traffic saturates the link. Such resiliency
may be developed several ways, but characteristics such as small
message size, asynchronous, redundant message delivery and minimal
connection overhead (when possible given local network policy) will
tend to contribute to the robustness demanded by a viable DOTS
protocol. Operators of peer DOTS-enabled domains may enable quality-
or class-of-service traffic tagging to increase the probability of
successful DOTS signal delivery, but DOTS requires no such policies
be in place. The DOTS solution indeed must be viable especially in
their absence.

On the other hand, DOTS must include protections ensuring message
confidentiality, integrity and authenticity to keep the protocol from
becoming another vector for the very attacks it's meant to help fight
off. DOTS clients must be able to authenticate DOTS servers, and
vice versa, for DOTS to operate safely, meaning the DOTS agents must
have a way to negotiate and agree upon the terms of protocol
security. Attacks against the transport protocol should not offer a
means of attack against the message confidentiality, integrity and
authenticity.

The DOTS server and client must also have some common method of
defining the scope of any mitigation performed by the mitigator, as
well as making adjustments to other commonly configurable features,
such as listen ports, exchanging black- and white-lists, and so on.

Finally, DOTS should provide sufficient extensibility to meet local,
vendor or future needs in coordinated attack defense, although this
consideration is necessarily superseded by the other operational
requirements.

2.1. General Requirements

   GEN-001   Extensibility: Protocols and data models developed as part
             of DOTS MUST be extensible in order to keep DOTS adaptable to
             operational and proprietary DDoS defenses. Future extensions MUST
             be backward compatible.

   GEN-002   Resilience and Robustness: The signaling protocol MUST be
             designed to maximize the probability of signal delivery even under
             the severely constrained network conditions imposed by particular
             attack traffic. The protocol MUST be resilient, that is, continue
             operating despite message loss and out-of-order or redundant
             message delivery. In support signaling protocol robustness, DOTS
             signals SHOULD be conveyed over a transport not susceptible to
             Head of Line Blocking.

   GEN-003   Bidirectionality: To support peer health detection, to
             maintain an open signal channel, and to increase the probability
             of signal delivery during attack, the signal channel MUST be
             bidirectional, with client and server transmitting signals to each
             other at regular intervals, regardless of any client request for
             mitigation. Unidirectional messages MUST be supported within the
             bidirectional signal channel to allow for unsolicited message
             delivery, enabling asynchronous notifications between agents.

   GEN-004   Sub-MTU Message Size: To avoid message fragmentation and the
             consequently decreased probability of message delivery, signaling
             protocol message size MUST be kept under signaling Path Maximum
             Transmission Unit (PMTU), including the byte overhead of any
             encapsulation, transport headers, and transport- or message-level
             security.

             DOTS agents SHOULD attempt to learn the PMTU through mechanisms
             such as Path MTU Discovery [RFC1191] or Packetization Layer Path
             MTU Discovery [RFC4821]. If the PMTU cannot be discovered, DOTS
             agents SHOULD assume a PMTU of 1280 bytes. If IPv4 support on
             legacy or otherwise unusual networks is a consideration and PMTU
             is unknown, DOTS implementations MAY rely on a PMTU of 576 bytes,
             as discussed in [RFC0791] and [RFC1122].

   GEN-005   Bulk Data Exchange: Infrequent bulk data exchange between
             DOTS agents can also significantly augment attack response
             coordination, permitting such tasks as population of black- or
             white-listed source addresses; address or prefix group aliasing;
             exchange of incident reports; and other hinting or configuration
             supplementing attack response.

As the resilience requirements for the DOTS signal channel mandate
small signal message size, a separate, secure data channel
utilizing a reliable transport protocol MUST be used for bulk data
exchange.

2.2. Operational Requirements

   OP-001 Use of Common Transport Protocols: DOTS MUST operate over
      common widely deployed and standardized transport protocols.
      While the User Datagram Protocol (UDP) [RFC0768] SHOULD be used
      for the signal channel, the Transmission Control Protocol (TCP)
      [RFC0793] MAY be used if necessary due to network policy or
      middlebox capabilities or configurations. The data channel MUST
      use a reliable transport; see Section 2.3 below.

   OP-002 Session Health Monitoring: Peer DOTS agents MUST regularly
      send heartbeats to each other after mutual authentication in order
      to keep the DOTS session active. A session MUST be considered
      active until a DOTS agent explicitly ends the session, or either
      DOTS agent fails to receive heartbeats from the other after a
      mutually agreed upon timeout period has elapsed.

   OP-003 Session Redirection: In order to increase DOTS operational
      flexibility and scalability, DOTS servers SHOULD be able to
      redirect DOTS clients to another DOTS server at any time. DOTS
      clients MUST NOT assume the redirection target DOTS server shares
      security state with the redirecting DOTS server. DOTS clients MAY
      attempt abbreviated security negotiation methods supported by the
      protocol, such as DTLS session resumption, but MUST be prepared to
      negotiate new security state with the redirection target DOTS
      server.

      Due to the increased likelihood of packet loss caused by link
      congestion during an attack, it is RECOMMENDED DOTS servers avoid
      redirecting while mitigation is enabled during an active attack
      against a target in the DOTS client's domain.

   OP-004 Mitigation Requests and Status: Authorized DOTS clients MUST
      be able to request scoped mitigation from DOTS servers. DOTS
      servers MUST send mitigation request status in response to DOTS
      clients requests for mitigation, and SHOULD accept scoped
      mitigation requests from authorized DOTS clients. DOTS servers
      MAY reject authorized requests for mitigation, but MUST include a
      reason for the rejection in the status message sent to the client.

      Due to the higher likelihood of packet loss during a DDoS attack,
      DOTS servers SHOULD regularly send mitigation status to authorized

DOTS clients which have requested and been granted mitigation, regardless of client requests for mitigation status.

When DOTS client-requested mitigation is active, DOTS server status messages SHOULD include the following mitigation metrics:

*   Total number of packets blocked by the mitigation

*   Current number of packets per second blocked

*   Total number of bytes blocked

*   Current number of bytes per second blocked

DOTS clients SHOULD take these metrics into account when determining whether to ask the DOTS server to cease mitigation.

Once a DOTS client requests mitigation, the client MAY withdraw that request at any time, regardless of whether mitigation is currently active. The DOTS server MUST immediately acknowledge a DOTS client's request to stop mitigation.

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 five minutes after acknowledging a DOTS client's withdrawal of a mitigation request. During this period, DOTS server status messages SHOULD indicate that mitigation is active but terminating. After the five-minute period elapses, the DOTS server MUST treat the mitigation as terminated, as the DOTS client is no longer responsible for the mitigation. For example, if there is a financial relationship between the DOTS client and server domains, the DOTS client ceases incurring cost at this point.

OP-005 Mitigation Lifetime: DOTS servers MUST support mitigation lifetimes, and MUST terminate a mitigation when the lifetime elapses. DOTS servers also MUST support renewal of mitigation lifetimes in mitigation requests from DOTS clients, allowing clients to extend mitigation as necessary for the duration of an attack.

DOTS servers MUST treat a mitigation terminated due to lifetime expiration exactly as if the DOTS client originating the mitigation had asked to end the mitigation, including the five-minute termination period, as described above in OP-004.

DOTS clients SHOULD include a mitigation lifetime in all
mitigation requests. If a DOTS client does not include a
mitigation lifetime in requests for help sent to the DOTS server,
the DOTS server will use a reasonable default as defined by the
protocol.

DOTS servers SHOULD support indefinite mitigation lifetimes,
enabling architectures in which the mitigator is always in the
traffic path to the resources for which the DOTS client is
requesting protection. DOTS servers MAY refuse mitigations with
indefinite lifetimes, for policy reasons. The reasons themselves
are out of scope for this document, but MUST be included in the
mitigation rejection message from the server, per OP-004.

OP-006 Mitigation Scope: DOTS clients MUST indicate desired
mitigation scope. The scope type will vary depending on the
resources requiring mitigation. All DOTS agent implementations
MUST support the following required scope types:

* IPv4 addresses in dotted quad format

* IPv4 address prefixes in CIDR notation [RFC4632]

* IPv6 addresses [RFC2373]

* IPv6 address prefixes [RFC2373]

* Domain names [RFC1035]

The following mitigation scope types are OPTIONAL:

* Uniform Resource Identifiers [RFC3986]

DOTS agents MUST support mitigation scope aliases, allowing DOTS
client and server to refer to collections of protected resources
by an opaque identifier created through the data channel, direct
configuration, or other means.

If there is additional information available narrowing the scope
of any requested attack response, such as targeted port range,
protocol, or service, DOTS clients SHOULD include that information
in client signals. DOTS clients MAY also include additional
attack details. Such supplemental information is OPTIONAL, and
DOTS servers MAY ignore it when enabling countermeasures on the
mitigator.

As an active attack evolves, clients MUST be able to adjust as
necessary the scope of requested mitigation by refining the scope
of resources requiring mitigation.

OP-007 Mitigation Efficacy: When a mitigation request by a DOTS
client is active, DOTS clients SHOULD transmit a metric of
perceived mitigation efficacy to the DOTS server, per "Automatic
or Operator-Assisted CPE or PE Mitigators Request Upstream DDoS
Mitigation Services" in [I-D.ietf-dots-use-cases]. DOTS servers
MAY use the efficacy metric to adjust countermeasures activated on
a mitigator on behalf of a DOTS client.

OP-008 Conflict Detection and Notification: Multiple DOTS clients
controlled by a single administrative entity may send conflicting
mitigation requests for pool of protected resources , as a result
of misconfiguration, operator error, or compromised DOTS clients.
DOTS servers attempting to honor conflicting requests may flap
network route or DNS information, degrading the networks
attempting to participate in attack response with the DOTS
clients. DOTS servers SHALL detect such conflicting requests, and
SHALL notify the DOTS clients in conflict. The notification
SHOULD indicate the nature and scope of the conflict, for example,
the overlapping prefix range in a conflicting mitigation request.

OP-009: Network Address Translator Traversal: The DOTS protocol MUST
operate over networks in which Network Address Translation (NAT)
is deployed. As UDP is the recommended transport for the DOTS
signal channel, all considerations in "Middlebox Traversal
Guidelines" in [RFC5405] apply to DOTS. Regardless of transport,
DOTS protocols MUST follow established best common practices
(BCPs) for NAT traversal.

2.3. Data Channel Requirements

The data channel is intended to be used for bulk data exchanges
between DOTS agents. Unlike the signal channel, which must operate
nominally even when confronted with signal degradation due to packet
loss, the data channel is not expected to be constructed to deal with
attack conditions. As the primary function of the data channel is
data exchange, a reliable transport is required in order for DOTS
agents to detect data delivery success or failure.

The data channel must be extensible. We anticipate the data channel
will be used for such purposes as configuration or resource
discovery. For example, a DOTS client may submit to the DOTS server
a collection of prefixes it wants to refer to by alias when
requesting mitigation, to which the server would respond with a
success status and the new prefix group alias, or an error status and

message in the event the DOTS client's data channel request failed.
The transactional nature of such data exchanges suggests a separate
set of requirements for the data channel, while the potentially
sensitive content sent between DOTS agents requires extra precautions
to ensure data privacy and authenticity.

DATA-001 Reliable transport: Messages sent over the data channel
   MUST be delivered reliably, in order sent.

DATA-002 Data privacy and integrity: Transmissions over the data
   channel are likely to contain operationally or privacy-sensitive
   information or instructions from the remote DOTS agent. Theft or
   modification of data channel transmissions could lead to
   information leaks or malicious transactions on behalf of the
   sending agent (see Section 4 below). Consequently data sent over
   the data channel MUST be encrypted and authenticated using current
   industry best practices. DOTS servers MUST enable means to
   prevent leaking operationally or privacy-sensitive data. Although
   administrative entities participating in DOTS may detail what data
   may be revealed to third-party DOTS agents, such considerations
   are not in scope for this document.

DATA-003 Resource Configuration: To help meet the general and
   operational requirements in this document, DOTS server
   implementations MUST provide an interface to configure resource
   identifiers, as described in OP-007. DOTS server implementations
   MAY expose additional configurability. Additional configurability
   is implementation-specific.

DATA-004 Black- and whitelist management: DOTS servers SHOULD
   provide methods for DOTS clients to manage black- and white-lists
   of traffic destined for resources belonging to a client.

   For example, a DOTS client should be able to create a black- or
   whitelist entry; retrieve a list of current entries from either
   list; update the content of either list; and delete entries as
   necessary.

   How the DOTS server determines client ownership of address space
   is not in scope.

2.4. Security requirements

   DOTS must operate within a particularly strict security context, as
   an insufficiently protected signal or data channel may be subject to
   abuse, enabling or supplementing the very attacks DOTS purports to
   mitigate.

SEC-001 Peer Mutual Authentication: DOTS agents MUST authenticate
   each other before a DOTS session is considered valid. The method
   of authentication is not specified, but should follow current
   industry best practices with respect to any cryptographic
   mechanisms to authenticate the remote peer.

SEC-002 Message Confidentiality, Integrity and Authenticity: DOTS
   protocols MUST take steps to protect the confidentiality,
   integrity and authenticity of messages sent between client and
   server. While specific transport- and message-level security
   options are not specified, the protocols MUST follow current
   industry best practices for encryption and message authentication.

   In order for DOTS protocols to remain secure despite advancements
   in cryptanalysis and traffic analysis, DOTS agents MUST be able to
   negotiate the terms and mechanisms of protocol security, subject
   to the interoperability and signal message size requirements
   above.

   While the interfaces between downstream DOTS server and upstream
   DOTS client within a DOTS gateway are implementation-specific,
   those interfaces nevertheless MUST provide security equivalent to
   that of the signaling sessions bridged by gateways in the
   signaling path. For example, when a DOTS gateway consisting of a
   DOTS server and DOTS client is running on the same logical device,
   they must be within the same process security boundary.

SEC-003 Message Replay Protection: In order to prevent a passive
   attacker from capturing and replaying old messages, DOTS protocols
   MUST provide a method for replay detection.

2.5. Data Model Requirements

   The value of DOTS is in standardizing a mechanism to permit elements,
   networks or domains under or under threat of DDoS attack to request
   aid mitigating the effects of any such attack. A well-structured
   DOTS data model is therefore critical to the development of a
   successful DOTS protocol.

DM-001: Structure: The data model structure for the DOTS protocol
   may be described by a single module, or be divided into related
   collections of hierarchical modules and sub-modules. If the data
   model structure is split across modules, those distinct modules
   MUST allow references to describe the overall data model's
   structural dependencies.

DM-002: Versioning: To ensure interoperability between DOTS protocol
   implementations, data models MUST be versioned. The version

number of the initial data model SHALL be 1. Each published
change to the initial published DOTS data model SHALL increment
the data model version by 1.

How the protocol represents data model versions is not defined in
this document.

DM-003: Mitigation Status Representation: The data model MUST
provide the ability to represent a request for mitigation and the
withdrawal of such a request. The data model MUST also support a
representation of currently requested mitigation status, including
failures and their causes.

DM-004: Mitigation Scope Representation: The data model MUST support
representation of a requested mitigation's scope. As mitigation
scope may be represented in several different ways, per OP-006
above, the data model MUST be capable of flexible representation
of mitigation scope.

DM-005: Mitigation Lifetime Representation: The data model MUST
support representation of a mitigation request's lifetime,
including mitigations with no specified end time.

DM-006: Mitigation Efficacy Representation: The data model MUST
support representation of a DOTS client's understanding of the
efficacy of a mitigation enabled through a mitigation request.

DM-007: Acceptable Signal Loss Representation: The data model MUST
be able to represent the DOTS agent's preference for acceptable
signal loss when establishing a signaling session, as described in
GEN-002.

DM-008: Heartbeat Interval Representation: The data model MUST be
able to represent the DOTS agent's preferred heartbeat interval,
which the client may include when establishing the signal channel,
as described in OP-002.

DM-009: Relationship to Transport: The DOTS data model MUST NOT
depend on the specifics of any transport to represent fields in
the model.

3. Congestion Control Considerations

3.1. Signal Channel

As part of a protocol expected to operate over links affected by DDoS
attack traffic, the DOTS signal channel MUST NOT contribute
significantly to link congestion. To meet the operational

requirements above, DOTS signal channel implementations MUST support
UDP. However, UDP when deployed naively can be a source of network
congestion, as discussed in [RFC5405]. Signal channel
implementations using UDP MUST therefore include a congestion control
mechanism.

Signal channel implementations using TCP may rely on built-in TCP
congestion control support.

3.2. Data Channel

As specified in DATA-001, the data channel requires reliable, in-
order message delivery. Data channel implementations using TCP may
rely on the TCP implementation's built-in congestion control
mechanisms.

4. Security Considerations

DOTS is at risk from three primary attacks:

o DOTS agent impersonation

o Traffic injection

o Signaling blocking

The DOTS protocol MUST be designed for minimal data transfer to
address the blocking risk. Impersonation and traffic injection
mitigation can be managed through current secure communications best
practices. See Section 2.4 above for a detailed discussion.

5. Contributors

Mohamed Boucadair
   Orange

   mohamed.boucadair@orange.com

Flemming Andreasen
   Cisco Systems, Inc.

   fandreas@cisco.com

Dave Dolson
   Sandvine

   ddolson@sandvine.com

6. Acknowledgments

    Thanks to Roman Danyliw and Matt Richardson for careful reading and
    feedback.

7. Change Log

7.1. 04 revision

    2017-03-13

    o Establish required and optional mitigation scope types

    o Specify message size for DOTS signal channel

    o Recast mitigation lifetime as a DOTS server requirement

    o Clarify DOTS server's responsibilities after client request to end
       mitigation

    o Specify security state handling on redirection

    o Signal channel should use transport not susceptible to HOL
       blocking

    o Expanded list of DDoS types to include network links

7.2. 03 revision

    2016-10-30

    o Extended SEC-003 to require secure interfaces within DOTS
       gateways.

    o Changed DATA-003 to Resource Configuration, delegating control of
       acceptable signal loss, heartbeat intervals, and mitigation
       lifetime to DOTS client.

    o Added data model requirements reflecting client control over the
       above.

7.3. 02 revision

7.4. 01 revision

    2016-03-21

   o Reconciled terminology with -00 revision of
      [I-D.ietf-dots-use-cases].

   o Terminology clarification based on working group feedback.

   o Moved security-related requirements to separate section.

   o Made resilience/robustness primary general requirement to align
      with charter.

   o Clarified support for unidirectional communication within the
      bidirectional signal channel.

   o Added proposed operational requirement to support session
      redirection.

   o Added proposed operational requirement to support conflict
      notification.

   o Added proposed operational requirement to support mitigation
      lifetime in mitigation requests.

   o Added proposed operational requirement to support mitigation
      efficacy reporting from DOTS clients.

   o Added proposed operational requirement to cache lookups of all
      kinds.

   o Added proposed operational requirement regarding NAT traversal.

   o Removed redundant mutual authentication requirement from data
      channel requirements.

7.5. 00 revision

   2015-10-15

7.6. Initial revision

   2015-09-24 Andrew Mortensen

8. References

8.1. Normative References

   [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768,
              DOI 10.17487/RFC0768, August 1980,
              <http://www.rfc-editor.org/info/rfc768>.

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

   [RFC0793] Postel, J., "Transmission Control Protocol", STD 7,
              RFC 793, DOI 10.17487/RFC0793, September 1981,
              <http://www.rfc-editor.org/info/rfc793>.

   [RFC1035] Mockapetris, P., "Domain names - implementation and
              specification", STD 13, RFC 1035, DOI 10.17487/RFC1035,
              November 1987, <http://www.rfc-editor.org/info/rfc1035>.

   [RFC1122] Braden, R., Ed., "Requirements for Internet Hosts -
              Communication Layers", STD 3, RFC 1122,
              DOI 10.17487/RFC1122, October 1989,
              <http://www.rfc-editor.org/info/rfc1122>.

   [RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191,
              DOI 10.17487/RFC1191, November 1990,
              <http://www.rfc-editor.org/info/rfc1191>.

   [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>.

   [RFC2373] Hinden, R. and S. Deering, "IP Version 6 Addressing
              Architecture", RFC 2373, DOI 10.17487/RFC2373, July 1998,
              <http://www.rfc-editor.org/info/rfc2373>.

   [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform
              Resource Identifier (URI): Generic Syntax", STD 66,
              RFC 3986, DOI 10.17487/RFC3986, January 2005,
              <http://www.rfc-editor.org/info/rfc3986>.

   [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>.

   [RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU
              Discovery", RFC 4821, DOI 10.17487/RFC4821, March 2007,
              <http://www.rfc-editor.org/info/rfc4821>.

   [RFC5405] Eggert, L. and G. Fairhurst, "Unicast UDP Usage Guidelines
              for Application Designers", RFC 5405,
              DOI 10.17487/RFC5405, November 2008,
              <http://www.rfc-editor.org/info/rfc5405>.

8.2. Informative References

   [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-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-03 (work
             in progress), November 2016.

   [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston,
             A., Peterson, J., Sparks, R., Handley, M., and E.
             Schooler, "SIP: Session Initiation Protocol", RFC 3261,
             DOI 10.17487/RFC3261, June 2002,
             <http://www.rfc-editor.org/info/rfc3261>.

   [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>.

Authors' Addresses

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

   Email: amortensen@arbor.net

   Robert Moskowitz
   HTT Consulting
   Oak Park, MI 42837
   United States

   Email: rgm@htt-consult.com

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

   Email: tireddy@cisco.com

      Distributed Denial of Service (DDoS) Open Threat Signaling Requirements
                     draft-ietf-dots-requirements-14

   Abstract

      This document defines the requirements for the Distributed Denial of
      Service (DDoS) Open Threat Signaling (DOTS) protocols enabling
      coordinated response to DDoS attacks.

   Status of This Memo

   Copyright Notice

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

Table of Contents

1. Introduction

1.1. Context and Motivation

   Distributed Denial of Service (DDoS) attacks afflict networks of all
   kinds, plaguing network operators at service providers and
   enterprises around the world. High-volume attacks saturating inbound
   links are now common, as attack scale and frequency continue to
   increase.

   The prevalence and impact of these DDoS attacks has led to an
   increased focus on coordinated attack response. However, many
   enterprises lack the resources or expertise to operate on-premises
   attack mitigation solutions themselves, or are constrained by local
   bandwidth limitations. To address such gaps, service providers have
   begun to offer on-demand traffic scrubbing services, which are
   designed to separate the DDoS attack traffic from legitimate traffic
   and forward only the latter.

   Today, these services offer proprietary interfaces for subscribers to
   request attack mitigation. Such proprietary interfaces tie a
   subscriber to a service while also limiting the network elements

capable of participating in the attack mitigation. As a result of
signaling interface incompatibility, attack responses may be
fragmented or otherwise incomplete, leaving operators in the attack
path unable to assist in the defense.

A standardized method to coordinate a real-time response among
involved operators will increase the speed and effectiveness of DDoS
attack mitigation, and reduce the impact of these attacks. This
document describes the required characteristics of protocols that
enable attack coordination and mitigation of DDoS attacks.

DDoS Open Threat Signaling (DOTS) communicates the need for defensive
action in anticipation of or in response to an attack, but does not
dictate the implementation of these actions. The requirements in
this document are derived from [I-D.ietf-dots-use-cases] and
[I-D.ietf-dots-architecture].

1.2. 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].

This document adopts the following terms:

DDoS: A distributed denial-of-service attack, in which traffic
   originating from multiple sources is directed at a target on a
   network. DDoS attacks are intended to cause a negative impact on
   the availability and/or other functionality of an attack target.
   Denial-of-service considerations are discussed in detail in
   [RFC4732].

DDoS attack target: A network connected entity with a finite set of
   resources, such as network bandwidth, memory or CPU, that is the
   target of a DDoS attack. Potential targets include (but are not
   limited to) network elements, network links, servers, and
   services.

DDoS attack telemetry: Collected measurements and behavioral
   characteristics defining the nature of a DDoS attack.

Countermeasure: An action or set of actions focused on recognizing
   and filtering out specific types of DDoS attack traffic while
   passing legitimate traffic to the attack target. Distinct
   countermeasures can be layered to defend against attacks combining
   multiple DDoS attack types.

   Mitigation: A set of countermeasures enforced against traffic
      destined for the target or targets of a detected or reported DDoS
      attack, where countermeasure enforcement is managed by an entity
      in the network path between attack sources and the attack target.
      Mitigation methodology is out of scope for this document.

   Mitigator: An entity, typically a network element, capable of
      performing mitigation of a detected or reported DDoS attack. The
      means by which this entity performs these mitigations and how they
      are requested of it are out of scope. The mitigator and DOTS
      server receiving a mitigation request are assumed to belong to the
      same administrative entity.

   DOTS client: A DOTS-aware software module responsible for requesting
      attack response coordination with other DOTS-aware elements.

   DOTS server: A DOTS-aware software module handling and responding to
      messages from DOTS clients. The DOTS server enables mitigation on
      behalf of the DOTS client, if requested, by communicating the DOTS
      client's request to the mitigator and returning selected mitigator
      feedback to the requesting DOTS client.

   DOTS agent: Any DOTS-aware software module capable of participating
      in a DOTS signal or data channel. It can be a DOTS client, DOTS
      server, or, as a logical agent, a DOTS gateway.

   DOTS gateway: A DOTS-aware software module resulting from the
      logical concatenation of the functionality of a DOTS server and a
      DOTS client into a single DOTS agent. This functionality is
      analogous to a Session Initiation Protocol (SIP) [RFC3261] Back-
      to-Back User Agent (B2BUA) [RFC7092]. A DOTS gateway has a
      client-facing side, which behaves as a DOTS server for downstream
      clients, and a server-facing side, which performs the role of DOTS
      client for upstream DOTS servers. Client-domain DOTS gateways are
      DOTS gateways that are in the DOTS client's domain, while server-
      domain DOTS gateways denote DOTS gateways that are in the DOTS
      server's domain. DOTS gateways are described further in
      [I-D.ietf-dots-architecture].

   Signal channel: A bidirectional, mutually authenticated
      communication channel between DOTS agents that is resilient even
      in conditions leading to severe packet loss, such as a volumetric
      DDoS attack causing network congestion.

   DOTS signal: A concise authenticated status/control message
      transmitted over the signal channel between DOTS agents, used to
      indicate the client's need for mitigation, as well as to convey
      the status of any requested mitigation.

Heartbeat: A message transmitted between DOTS agents over the signal
   channel, used as a keep-alive and to measure peer health.

Data channel: A bidirectional, mutually authentication
   communincation channel between two DOTS agents used for infrequent
   but reliable bulk exchange of data not easily or appropriately
   communicated through the signal channel under attack conditions.

Filter: A specification of a matching network traffic flow or set of
   flows. The filter will typically have a policy associated with
   it, e.g., rate-limiting or discarding matching traffic [RFC4949].

Blacklist: A list of filters indicating sources from which traffic
   should be blocked, regardless of traffic content.

Whitelist: A list of filters indicating sources from which traffic
   should always be allowed, regardless of contradictory data gleaned
   in a detected attack.

Multi-homed DOTS client: A DOTS client exchanging messages with
   multiple DOTS servers, each in a separate administrative domain.

2. Requirements

   This section describes the required features and characteristics of
   the DOTS protocols.

   The DOTS protocols enable and manage mitigation on behalf of a
   network domain or resource which is or may become the focus of a DDoS
   attack. An active DDoS attack against the entity controlling the
   DOTS client need not be present before establishing a communication
   channel between DOTS agents. Indeed, establishing a relationship
   with peer DOTS agents during normal network conditions provides the
   foundation for more rapid attack response against future attacks, as
   all interactions setting up DOTS, including any business or service
   level agreements, are already complete. Reachability information of
   peer DOTS agents is provisioned to a DOTS client using a variety of
   manual or dynamic methods. Once a relationship between DOTS agents
   is established, regular communication between DOTS clients and
   servers enables a common understanding of the DOTS agents' health and
   activity.

   The DOTS protocol must at a minimum make it possible for a DOTS
   client to request aid mounting a defense, coordinated by a DOTS
   server, against a suspected attack, signaling within or between
   domains as requested by local operators. DOTS clients should
   similarly be able to withdraw aid requests. DOTS requires no
   justification from DOTS clients for requests for help, nor do DOTS

clients need to justify withdrawing help requests: the decision is
local to the DOTS clients' domain. Multi-homed DOTS clients must be
able to select the appropriate DOTS server(s) to which a mitigation
request is to be sent. The method for selecting the appropriate DOTS
server in a multi-homed environment is out of scope.

DOTS protocol implementations face competing operational goals when
maintaining this bidirectional communication stream. On the one
hand, DOTS must include protections ensuring message confidentiality,
integrity and authenticity to keep the protocols from becoming
additional vectors for the very attacks it is meant to help fight
off. On the other hand, the protocol must be resilient under
extremely hostile network conditions, providing continued contact
between DOTS agents even as attack traffic saturates the link. Such
resiliency may be developed several ways, but characteristics such as
small message size, asynchronous, redundant message delivery and
minimal connection overhead (when possible given local network
policy) will tend to contribute to the robustness demanded by a
viable DOTS protocol. Operators of peer DOTS-enabled domains may
enable quality- or class-of-service traffic tagging to increase the
probability of successful DOTS signal delivery, but DOTS does not
require such policies be in place, and should be viable in their
absence.

The DOTS server and client must also have some standardized method of
defining the scope of any mitigation, as well as managing other
mitigation-related configuration.

Finally, DOTS should be sufficiently extensible to meet future needs
in coordinated attack defense, although this consideration is
necessarily superseded by the other operational requirements.

2.1. General Requirements

   GEN-001 Extensibility: Protocols and data models developed as part
      of DOTS MUST be extensible in order to keep DOTS adaptable to
      operational and proprietary DDoS defenses. Future extensions MUST
      be backward compatible. DOTS protocols MUST use a version number
      system to distinguish protocol revisions. Implementations of
      older protocol versions SHOULD ignore information added to DOTS
      messages as part of newer protocol versions.

   GEN-002 Resilience and Robustness: The signaling protocol MUST be
      designed to maximize the probability of signal delivery even under
      the severely constrained network conditions caused by particular
      attack traffic. The protocol MUST be resilient, that is, continue
      operating despite message loss and out-of-order or redundant
      message delivery. In support of signaling protocol robustness,

DOTS signals SHOULD be conveyed over a transport not susceptible
to Head of Line Blocking.

GEN-003 Bulk Data Exchange: Infrequent bulk data exchange between
DOTS agents can also significantly augment attack response
coordination, permitting such tasks as population of black- or
white-listed source addresses; address or prefix group aliasing;
exchange of incident reports; and other hinting or configuration
supplementing attack response.

As the resilience requirements for the DOTS signal channel mandate
small signal message size, a separate, secure data channel
utilizing a reliable transport protocol MUST be used for bulk data
exchange.

GEN-004 Mitigation Hinting: DOTS clients may have access to attack
details which can be used to inform mitigation techniques.
Example attack details might include locally collected
fingerprints for an on-going attack, or anticipated or active
attack focal points based on other threat intelligence. DOTS
clients MAY send mitigation hints derived from attack details to
DOTS servers, in the full understanding that the DOTS server MAY
ignore mitigation hints. Mitigation hints MAY be transmitted
across either signal or data channel. DOTS server handling of
mitigation hints is implementation-specific.

GEN-005 Loop Handling: In certain scenarios, typically involving
misconfiguration of DNS or routing policy, it may be possible for
communication between DOTS agents to loop. Signal and data
channel implementations should be prepared to detect and terminate
such loops to prevent service disruption.

2.2. Signal Channel Requirements

SIG-001 Use of Common Transport Protocols: DOTS MUST operate over
common widely deployed and standardized transport protocols.
While connectionless transport such as the User Datagram Protocol
(UDP) [RFC0768] SHOULD be used for the signal channel, the
Transmission Control Protocol (TCP) [RFC0793] MAY be used if
necessary due to network policy or middlebox capabilities or
configurations.

SIG-002 Sub-MTU Message Size: To avoid message fragmentation and the
consequently decreased probability of message delivery over a
congested link, signaling protocol message size MUST be kept under
signaling Path Maximum Transmission Unit (PMTU), including the
byte overhead of any encapsulation, transport headers, and
transport- or message-level security.

DOTS agents SHOULD attempt to learn the PMTU through mechanisms
such as Path MTU Discovery [RFC1191] or Packetization Layer Path
MTU Discovery [RFC4821]. If the PMTU cannot be discovered, DOTS
agents SHOULD assume a PMTU of 1280 bytes. If IPv4 support on
legacy or otherwise unusual networks is a consideration and PMTU
is unknown, DOTS implementations MAY rely on a PMTU of 576 bytes,
as discussed in [RFC0791] and [RFC1122].

SIG-003 Bidirectionality: To support peer health detection, to
   maintain an active signal channel, and increase the probability of
   signal delivery during an attack, the signal channel MUST be
   bidirectional, with client and server transmitting signals to each
   other at regular intervals, regardless of any client request for
   mitigation. Unidirectional messages MUST be supported within the
   bidirectional signal channel to allow for unsolicited message
   delivery, enabling asynchronous notifications between DOTS agents.

SIG-004 Channel Health Monitoring: DOTS agents MUST support exchange
   of heartbeat messages over the signal channel to monitor channel
   health. Peer DOTS agents SHOULD regularly send heartbeats to each
   other while a mitigation request is active. The heartbeat
   interval during active mitigation could be negotiable, but SHOULD
   be frequent enough to maintain any on-path NAT or Firewall
   bindings during mitigation.

   To support scenarios in which loss of heartbeat is used to trigger
   mitigation, and to keep the channel active, DOTS clients MAY
   solicit heartbeat exchanges after successful mutual
   authentication. When DOTS agents are exchanging heartbeats and no
   mitigation request is active, either agent MAY request changes to
   the heartbeat rate. For example, a DOTS server might want to
   reduce heartbeat frequency or cease heartbeat exchanges when an
   active DOTS client has not requested mitigation, in order to
   control load.

   Following mutual authentication, a signal channel MUST be
   considered active until a DOTS agent explicitly ends the session,
   or either DOTS agent fails to receive heartbeats from the other
   after a mutually agreed upon retransmission procedure has been
   exhausted. Because heartbeat loss is much more likely during
   volumetric attack, DOTS agents SHOULD avoid signal channel
   termination when mitigation is active and heartbeats are not
   received by either DOTS agent for an extended period. In such
   circumstances, DOTS clients MAY attempt to reestablish the signal
   channel, but SHOULD continue to send heartbeats so that the DOTS
   server knows the session is still alive. DOTS servers are assumed
   to have the ability to monitor the attack, using feedback from the
   mitigator and other available sources, and MAY use the absence of

attack traffic and lack of client heartbeats as an indication the
signal channel is defunct.

SIG-005 Channel Redirection: In order to increase DOTS operational
    flexibility and scalability, DOTS servers SHOULD be able to
    redirect DOTS clients to another DOTS server at any time. DOTS
    clients MUST NOT assume the redirection target DOTS server shares
    security state with the redirecting DOTS server. DOTS clients are
    free to attempt abbreviated security negotiation methods supported
    by the protocol, such as DTLS session resumption, but MUST be
    prepared to negotiate new security state with the redirection
    target DOTS server.

    Due to the increased likelihood of packet loss caused by link
    congestion during an attack, DOTS servers SHOULD NOT redirect
    while mitigation is enabled during an active attack against a
    target in the DOTS client's domain.

SIG-006 Mitigation Requests and Status: Authorized DOTS clients MUST
    be able to request scoped mitigation from DOTS servers. DOTS
    servers MUST send status to the DOTS clients about mitigation
    requests. If a DOTS server rejects an authorized request for
    mitigation, the DOTS server MUST include a reason for the
    rejection in the status message sent to the client.

    Due to the higher likelihood of packet loss during a DDoS attack,
    DOTS servers SHOULD regularly send mitigation status to authorized
    DOTS clients which have requested and been granted mitigation,
    regardless of client requests for mitigation status.

    When DOTS client-requested mitigation is active, DOTS server
    status messages SHOULD include the following mitigation metrics:

    * Total number of packets blocked by the mitigation

    * Current number of packets per second blocked

    * Total number of bytes blocked

    * Current number of bytes per second blocked

    DOTS clients MAY take these metrics into account when determining
    whether to ask the DOTS server to cease mitigation.

    A DOTS client MAY withdraw a mitigation request at any time,
    regardless of whether mitigation is currently active. The DOTS
    server MUST immediately acknowledge a DOTS client's request to
    stop mitigation.

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 allow mitigation to continue
for a limited period after acknowledging a DOTS client's
withdrawal of a mitigation request. During this period, DOTS
server status messages SHOULD indicate that mitigation is active
but terminating.

The initial active-but-terminating period is implementation- and
deployment- specific, but SHOULD be sufficiently long to absorb
latency incurred by route propagation. If the client requests
mitigation again before the initial active-but-terminating period
elapses, the DOTS server MAY exponentially increase the active-
but-terminating period up to a maximum of 300 seconds (5 minutes).
After the active-but-terminating period elapses, the DOTS server
MUST treat the mitigation as terminated, as the DOTS client is no
longer responsible for the mitigation.

SIG-007 Mitigation Lifetime: DOTS servers MUST support mitigations
    for a negotiated time interval, and MUST terminate a mitigation
    when the lifetime elapses. DOTS servers also MUST support renewal
    of mitigation lifetimes in mitigation requests from DOTS clients,
    allowing clients to extend mitigation as necessary for the
    duration of an attack.

    DOTS servers MUST treat a mitigation terminated due to lifetime
    expiration exactly as if the DOTS client originating the
    mitigation had asked to end the mitigation, including the active-
    but-terminating period, as described above in SIG-005.

    DOTS clients MUST include a mitigation lifetime in all mitigation
    requests.

    DOTS servers SHOULD support indefinite mitigation lifetimes,
    enabling architectures in which the mitigator is always in the
    traffic path to the resources for which the DOTS client is
    requesting protection. DOTS clients MUST be prepared to not be
    granted mitigations with indefinite lifetimes. DOTS servers MAY
    refuse mitigations with indefinite lifetimes, for policy reasons.
    The reasons themselves are out of scope. If the DOTS server does
    not grant a mitigation request with an indefinite mitigation
    lifetime, it MUST set the lifetime to a value that is configured
    locally. That value MUST be returned in a reply to the requesting
    DOTS client.

SIG-008 Mitigation Scope: DOTS clients MUST indicate desired
    mitigation scope. The scope type will vary depending on the

resources requiring mitigation. All DOTS agent implementations
MUST support the following required scope types:

* IPv4 prefixes in CIDR notation [RFC4632]

* IPv6 prefixes [RFC4291][RFC5952]

* Domain names [RFC1035]

The following mitigation scope types are OPTIONAL:

* Uniform Resource Identifiers [RFC3986]

DOTS servers MUST be able to resolve domain names and (when
supported) URIs. How name resolution is managed on the DOTS
server is implementation-specific.

DOTS agents MUST support mitigation scope aliases, allowing DOTS
clients and servers to refer to collections of protected resources
by an opaque identifier created through the data channel, direct
configuration, or other means. Domain name and URI mitigation
scopes may be thought of as a form of scope alias, in which the
addresses to which the domain name or URI resolve represent the
full scope of the mitigation.

If there is additional information available narrowing the scope
of any requested attack response, such as targeted port range,
protocol, or service, DOTS clients SHOULD include that information
in client mitigation requests. DOTS clients MAY also include
additional attack details. DOTS servers MAY ignore such
supplemental information when enabling countermeasures on the
mitigator.

As an active attack evolves, DOTS clients MUST be able to adjust
as necessary the scope of requested mitigation by refining the
scope of resources requiring mitigation.

A DOTS client may obtain the mitigation scope through direct
provisioning or through implementation-specific methods of
discovery. DOTS clients MUST support at least one mechanism to
obtain mitigation scope.

SIG-009 Mitigation Efficacy: When a mitigation request is active,
   DOTS clients SHOULD transmit a metric of perceived mitigation
   efficacy to the DOTS server. DOTS servers MAY use the efficacy
   metric to adjust countermeasures activated on a mitigator on
   behalf of a DOTS client.

   SIG-010 Conflict Detection and Notification: Multiple DOTS clients
      controlled by a single administrative entity may send conflicting
      mitigation requests as a result of misconfiguration, operator
      error, or compromised DOTS clients. DOTS servers in the same
      administrative domain attempting to honor conflicting requests may
      flap network route or DNS information, degrading the networks
      attempting to participate in attack response with the DOTS
      clients. DOTS servers in a single administrative domain SHALL
      detect such conflicting requests, and SHALL notify the DOTS
      clients in conflict. The notification SHOULD indicate the nature
      and scope of the conflict, for example, the overlapping prefix
      range in a conflicting mitigation request.

   SIG-011 Network Address Translator Traversal: DOTS clients may be
      deployed behind a Network Address Translator (NAT), and need to
      communicate with DOTS servers through the NAT. DOTS protocols
      MUST therefore be capable of traversing NATs.

      If UDP is used as the transport for the DOTS signal channel, all
      considerations in "Middlebox Traversal Guidelines" in [RFC8085]
      apply to DOTS. Regardless of transport, DOTS protocols MUST
      follow established best common practices established in BCP 127
      for NAT traversal [RFC4787][RFC6888][RFC7857].

2.3. Data Channel Requirements

   The data channel is intended to be used for bulk data exchanges
   between DOTS agents. Unlike the signal channel, the data channel is
   not expected to be constructed to deal with attack conditions. As
   the primary function of the data channel is data exchange, a reliable
   transport is required in order for DOTS agents to detect data
   delivery success or failure.

   The data channel provides a protocol for DOTS configuration,
   management. For example, a DOTS client may submit to a DOTS server a
   collection of prefixes it wants to refer to by alias when requesting
   mitigation, to which the server would respond with a success status
   and the new prefix group alias, or an error status and message in the
   event the DOTS client's data channel request failed.

   DATA-001 Reliable transport: Messages sent over the data channel
      MUST be delivered reliably, in order sent.

   DATA-002 Data privacy and integrity: Transmissions over the data
      channel are likely to contain operationally or privacy-sensitive
      information or instructions from the remote DOTS agent. Theft or
      modification of data channel transmissions could lead to
      information leaks or malicious transactions on behalf of the

sending agent (see Section 4 below). Consequently data sent over
the data channel MUST be encrypted and authenticated using current
IETF best practices. DOTS servers MUST enable means to prevent
leaking operationally or privacy-sensitive data. Although
administrative entities participating in DOTS may detail what data
may be revealed to third-party DOTS agents, such considerations
are not in scope for this document.

DATA-003 Resource Configuration: To help meet the general and signal
channel requirements in Section 2.1 and Section 2.2, DOTS server
implementations MUST provide an interface to configure resource
identifiers, as described in SIG-007. DOTS server implementations
MAY expose additional configurability. Additional configurability
is implementation-specific.

DATA-004 Black- and whitelist management: DOTS servers MUST provide
methods for DOTS clients to manage black- and white-lists of
traffic destined for resources belonging to a client.

For example, a DOTS client should be able to create a black- or
whitelist entry, retrieve a list of current entries from either
list, update the content of either list, and delete entries as
necessary.

How a DOTS server authorizes DOTS client management of black- and
white-list entries is implementation-specific.

2.4. Security Requirements

DOTS must operate within a particularly strict security context, as
an insufficiently protected signal or data channel may be subject to
abuse, enabling or supplementing the very attacks DOTS purports to
mitigate.

SEC-001 Peer Mutual Authentication: DOTS agents MUST authenticate
each other before a DOTS signal or data channel is considered
valid. The method of authentication is not specified, but should
follow current industry best practices with respect to any
cryptographic mechanisms to authenticate the remote peer.

SEC-002 Message Confidentiality, Integrity and Authenticity: DOTS
protocols MUST take steps to protect the confidentiality,
integrity and authenticity of messages sent between client and
server. While specific transport- and message-level security
options are not specified, the protocols MUST follow current
industry best practices for encryption and message authentication.

In order for DOTS protocols to remain secure despite advancements
in cryptanalysis and traffic analysis, DOTS agents MUST be able to
negotiate the terms and mechanisms of protocol security, subject
to the interoperability and signal message size requirements in
Section 2.2.

While the interfaces between downstream DOTS server and upstream
DOTS client within a DOTS gateway are implementation-specific,
those interfaces nevertheless MUST provide security equivalent to
that of the signal channels bridged by gateways in the signaling
path. For example, when a DOTS gateway consisting of a DOTS
server and DOTS client is running on the same logical device, the
two DOTS agents could be implemented within the same process
security boundary.

SEC-003 Message Replay Protection: To prevent a passive attacker
from capturing and replaying old messages, and thereby potentially
disrupting or influencing the network policy of the receiving DOTS
agent's domain, DOTS protocols MUST provide a method for replay
detection and prevention.

Within the signal channel, messages MUST be uniquely identified
such that replayed or duplicated messages can be detected and
discarded. Unique mitigation requests MUST be processed at most
once.

SEC-004 Authorization: DOTS servers MUST authorize all messages from
DOTS clients which pertain to mitigation, configuration,
filtering, or status.

DOTS servers MUST reject mitigation requests with scopes which the
DOTS client is not authorized to manage.

Likewise, DOTS servers MUST refuse to allow creation, modification
or deletion of scope aliases and black-/white-lists when the DOTS
client is unauthorized.

The modes of authorization are implementation-specific.

2.5. Data Model Requirements

A well-structured DOTS data model is critical to the development of
successful DOTS protocols.

DM-001 Structure: The data model structure for the DOTS protocol MAY
be described by a single module, or be divided into related
collections of hierarchical modules and sub-modules. If the data
model structure is split across modules, those distinct modules

      MUST allow references to describe the overall data model's
      structural dependencies.

   DM-002 Versioning: To ensure interoperability between DOTS protocol
      implementations, data models MUST be versioned. How the protocols
      represent data model versions is not defined in this document.

   DM-003 Mitigation Status Representation: The data model MUST provide
      the ability to represent a request for mitigation and the
      withdrawal of such a request. The data model MUST also support a
      representation of currently requested mitigation status, including
      failures and their causes.

   DM-004 Mitigation Scope Representation: The data model MUST support
      representation of a requested mitigation's scope. As mitigation
      scope may be represented in several different ways, per SIG-007
      above, the data model MUST be capable of flexible representation
      of mitigation scope.

   DM-005 Mitigation Lifetime Representation: The data model MUST
      support representation of a mitigation request's lifetime,
      including mitigations with no specified end time.

   DM-006 Mitigation Efficacy Representation: The data model MUST
      support representation of a DOTS client's understanding of the
      efficacy of a mitigation enabled through a mitigation request.

   DM-007 Acceptable Signal Loss Representation: The data model MUST be
      able to represent the DOTS agent's preference for acceptable
      signal loss when establishing a signal channel, as described in
      GEN-002.

   DM-008 Heartbeat Interval Representation: The data model MUST be
      able to represent the DOTS agent's preferred heartbeat interval,
      which the client may include when establishing the signal channel,
      as described in SIG-003.

   DM-009 Relationship to Transport: The DOTS data model MUST NOT
      depend on the specifics of any transport to represent fields in
      the model.

3. Congestion Control Considerations

3.1. Signal Channel

   As part of a protocol expected to operate over links affected by DDoS
   attack traffic, the DOTS signal channel MUST NOT contribute
   significantly to link congestion. To meet the signal channel

requirements above, DOTS signal channel implementations SHOULD
support connectionless transports. However, some connectionless
transports when deployed naively can be a source of network
congestion, as discussed in [RFC8085]. Signal channel
implementations using such connectionless transports, such as UDP,
therefore MUST include a congestion control mechanism.

Signal channel implementations using TCP may rely on built-in TCP
congestion control support.

3.2. Data Channel

As specified in DATA-001, the data channel requires reliable, in-
order message delivery. Data channel implementations using TCP may
rely on the TCP implementation's built-in congestion control
mechanisms.

4. Security Considerations

This document informs future protocols under development, and so does
not have security considerations of its own. However, operators
should be aware of potential risks involved in deploying DOTS. DOTS
agent impersonation and signal blocking are discussed here.
Additional DOTS security considerations may be found in
[I-D.ietf-dots-architecture] and DOTS protocol documents.

Impersonation of either DOTS server or DOTS client could have
catastrophic impact on operations in either domain. If an attacker
has the ability to impersonate a DOTS client, that attacker can
affect policy on the network path to the DOTS client's domain, up to
and including instantiation of blacklists blocking all inbound
traffic to networks for which the DOTS client is authorized to
request mitigation.

Similarly, an impersonated DOTS server may be able to act as a sort
of malicious DOTS gateway, intercepting requests from the downstream
DOTS client, and modifying them before transmission to the DOTS
server to inflict the desired impact on traffic to or from the DOTS
client's domain. Among other things, this malicious DOTS gateway
might receive and discard mitigation requests from the DOTS client,
ensuring no requested mitigation is ever applied.

As detailed in Section 2.4, DOTS implementations require mutual
authentication of DOTS agents in order to make agent impersonation
more difficult. However, impersonation may still be possible as a
result of credential theft, implementation flaws, or compromise of
DOTS agents. To detect misuse, DOTS operators should carefully

monitor and audit DOTS agents, while employing current secure network
communications best practices to reduce attack surface.

Blocking communication between DOTS agents has the potential to
disrupt the core function of DOTS, which is to request mitigation of
active or expected DDoS attacks. The DOTS signal channel is expected
to operate over congested inbound links, and, as described in
Section 2.2, the signal channel protocol must be designed for minimal
data transfer to reduce the incidence of signal blocking.

5. IANA Considerations

   This document does not require any IANA action.

6. Contributors

   Mohamed Boucadair
      Orange

      mohamed.boucadair@orange.com

   Flemming Andreasen
      Cisco Systems, Inc.

      fandreas@cisco.com

   Dave Dolson
      Sandvine

      ddolson@sandvine.com

7. Acknowledgments

   Thanks to Roman Danyliw and Matt Richardson for careful reading and
   feedback.

8. References

8.1. Normative References

   [RFC0768] Postel, J., "User Datagram Protocol", STD 6, RFC 768,
              DOI 10.17487/RFC0768, August 1980,
              <https://www.rfc-editor.org/info/rfc768>.

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

   [RFC0793] Postel, J., "Transmission Control Protocol", STD 7,
              RFC 793, DOI 10.17487/RFC0793, September 1981,
              <https://www.rfc-editor.org/info/rfc793>.

   [RFC1035] Mockapetris, P., "Domain names - implementation and
              specification", STD 13, RFC 1035, DOI 10.17487/RFC1035,
              November 1987, <https://www.rfc-editor.org/info/rfc1035>.

   [RFC1122] Braden, R., Ed., "Requirements for Internet Hosts -
              Communication Layers", STD 3, RFC 1122,
              DOI 10.17487/RFC1122, October 1989,
              <https://www.rfc-editor.org/info/rfc1122>.

   [RFC1191] Mogul, J. and S. Deering, "Path MTU discovery", RFC 1191,
              DOI 10.17487/RFC1191, November 1990,
              <https://www.rfc-editor.org/info/rfc1191>.

   [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
              Requirement Levels", BCP 14, RFC 2119,
              DOI 10.17487/RFC2119, March 1997,
              <https://www.rfc-editor.org/info/rfc2119>.

   [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform
              Resource Identifier (URI): Generic Syntax", STD 66,
              RFC 3986, DOI 10.17487/RFC3986, January 2005,
              <https://www.rfc-editor.org/info/rfc3986>.

   [RFC4291] Hinden, R. and S. Deering, "IP Version 6 Addressing
              Architecture", RFC 4291, DOI 10.17487/RFC4291, February
              2006, <https://www.rfc-editor.org/info/rfc4291>.

   [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, <https://www.rfc-editor.org/info/rfc4632>.

   [RFC4787] Audet, F., Ed. and C. Jennings, "Network Address
              Translation (NAT) Behavioral Requirements for Unicast
              UDP", BCP 127, RFC 4787, DOI 10.17487/RFC4787, January
              2007, <https://www.rfc-editor.org/info/rfc4787>.

   [RFC4821] Mathis, M. and J. Heffner, "Packetization Layer Path MTU
              Discovery", RFC 4821, DOI 10.17487/RFC4821, March 2007,
              <https://www.rfc-editor.org/info/rfc4821>.

   [RFC6888] Perreault, S., Ed., Yamagata, I., Miyakawa, S., Nakagawa,
              A., and H. Ashida, "Common Requirements for Carrier-Grade
              NATs (CGNs)", BCP 127, RFC 6888, DOI 10.17487/RFC6888,
              April 2013, <https://www.rfc-editor.org/info/rfc6888>.

   [RFC7857] Penno, R., Perreault, S., Boucadair, M., Ed., Sivakumar,
              S., and K. Naito, "Updates to Network Address Translation
              (NAT) Behavioral Requirements", BCP 127, RFC 7857,
              DOI 10.17487/RFC7857, April 2016,
              <https://www.rfc-editor.org/info/rfc7857>.

   [RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage
              Guidelines", BCP 145, RFC 8085, DOI 10.17487/RFC8085,
              March 2017, <https://www.rfc-editor.org/info/rfc8085>.

   [RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6
              Address Text Representation", RFC 5952,
              DOI 10.17487/RFC5952, August 2010,
              <https://www.rfc-editor.org/info/rfc5952>.

8.2. Informative References

   [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-05 (work in progress), October 2017.

   [I-D.ietf-dots-use-cases]
              Dobbins, R., Migault, D., Fouant, S., Moskowitz, R.,
              Teague, N., Xia, L., and K. Nishizuka, "Use cases for DDoS
              Open Threat Signaling", draft-ietf-dots-use-cases-09 (work
              in progress), November 2017.

   [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston,
              A., Peterson, J., Sparks, R., Handley, M., and E.
              Schooler, "SIP: Session Initiation Protocol", RFC 3261,
              DOI 10.17487/RFC3261, June 2002,
              <https://www.rfc-editor.org/info/rfc3261>.

   [RFC7092] Kaplan, H. and V. Pascual, "A Taxonomy of Session
              Initiation Protocol (SIP) Back-to-Back User Agents",
              RFC 7092, DOI 10.17487/RFC7092, December 2013,
              <https://www.rfc-editor.org/info/rfc7092>.

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

   [RFC4949] Shirey, R., "Internet Security Glossary, Version 2",
              FYI 36, RFC 4949, DOI 10.17487/RFC4949, August 2007,
              <https://www.rfc-editor.org/info/rfc4949>.

Authors' Addresses

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

   Email: amortensen@arbor.net

   Robert Moskowitz
   Huawei
   Oak Park, MI 42837
   United States

   Email: rgm@htt-consult.com

   Tirumaleswar Reddy
   McAfee
   Embassy Golf Link Business Park
   Bangalore, Karnataka 560071
   India

   Email: TirumaleswarReddy_Konda@McAfee.com

DOTS WG   R. Dobbins, Ed.
Internet-Draft   Arbor Networks
Intended status: Informational   S. Fouant
Expires: September 28, 2017

                                                         D. Migault
                                                           Ericsson
                                                       R. Moskowitz
                                                     HTT Consulting
                                                          N. Teague
                                                       Verisign Inc
                                                            L. Xia
                                                            Huawei
                                                       K. Nishizuka
                                                 NTT Communications
                                                     March 27, 2017

                    Use cases for DDoS Open Threat Signaling
                       draft-ietf-dots-use-cases-04.txt

Abstract

   The DDoS Open Threat Signaling (DOTS) effort is intended to provide a
   protocol that facilitates interoperability between multivendor
   solutions/services. This document presents use cases to evaluate the
   interactions expected between the DOTS components as well as the DOTS
   exchanges. The purpose of the use cases is to identify the
   interacting DOTS component, how they collaborate and what are the
   types of information to be exchanged.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

   Currently, distributed denial-of-service (DDoS) attack mitigation
   solutions/services are largely based upon siloed, proprietary
   communications paradigms which result in vendor/service lock-in. As
   a side-effect, this makes the configuration, provisioning, operation,
   and activation of these solutions a highly manual and often time-
   consuming process. Additionally, coordination of multiple DDoS
   mitigation solutions/services simultaneously engaged in defending the
   same organization against DDoS attacks is fraught with both technical

and process-related hurdles. This greatly increase operational
complexity and often results in suboptimal DDoS attack mitigation
efficacy.

The DDoS Open Threat Signaling (DOTS) effort is intended to provide a
protocol that facilitates interoperability between multivendor DDoS
mitigation solutions/services. As DDoS solutions/services are
broadly heterogeneous among different vendors, the primary goal for
DOTS is to provide a high level interaction with these DDoS
solutions/services such as initiating or terminating DDoS mitigation
assistance.

It should be noted that DOTS is not in and of itself intended to
perform orchestration functions duplicative of the functionality
being developed by the [I2NSF] WG; rather, DOTS is intended to allow
devices, services, and applications to request DDoS attack mitigation
assistance and receive mitigation status updates from systems of this
nature.

The use cases presented in the document are intended to provide
examples of communications interactions DOTS-enabled nodes in both
inter- and intra-organizational DDoS mitigation scenarios. These use
cases are expected to provide inputs for the design of the DOTS
protocol(s).

2. Terminology and Acronyms

2.1. Requirements 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 RFC 2119 [RFC2119].

2.2. Acronyms

This document makes use of the same terminology and definitions as
[I-D.ietf-dots-requirements], except where noted.

2.3. Terms

Inter-organizational: a DOTS communications relationship between
distinct organizations with separate spans of administrative control.
Typical inter-organizational DOTS communication relationships would
be between a DDoS mitigation service provider and an end-customer
organizational which requires DDoS mitigation assistance; between
multiple DDoS mitigation service providers coordinating mutual
defense of a mutual end-customer; or between DDoS mitigation service
providers which are requesting additional DDoS mitigation assistance

in for attacks which exceed their inherent DDoS mitigation capacities and/or capabilities.

Intra-organizational: a DOTS communications relationship between various elements within a single span of administrative control. A typical intra-organizational DOTS communications relationship would be between DOTS clients, DOTS gateways, and DOTS servers within the same organization.

3. Use Cases Scenarios

   This section provides a high-level description of scenarios addressed by DOTS. In both sections, the scenarios are provided in order to illustrate the use of DOTS in typical DDoS attack scenarios. They are not definitive, and other use cases are expected to emerge with widespread DOTS deployment.

   All scenarios present a coordination between the targeted organization, the DDoS attack telemetry and the mitigator. The coordination and communication between these entity depends, for example on the characteristic or functionality of the equipment, the reliability of the information provided by DDoS attack telemetry, and the business relationship between the DDoS target domain and the mitigator.

   More explicitly, in some cases, the DDoS attack telemetry may simply activate a DDoS mitigation, whereas in other cases, it may collaborate by providing some information about an attack. In some cases, the DDoS mitigation may be orchestrated, which includes selecting a specific appliance as well as starting/ending a mitigation.

3.1. Inter-domain Use Cases

3.1.1. Enterprise with an upsteam transit provider DDoS mitigation
        Service

   In this scenario, an enterprise network with self-hosted Internet-facing properties such as Web servers, authoritative DNS servers, and VoIP PBXes has an intelligent DDoS mitigation system (IDMS) deployed to protect those servers and applications from DDoS attacks. In addition to their on-premise DDoS defense capability, they have contracted with their Internet transit provider for DDoS mitigation services which threaten to overwhelm their transit link bandwidth.

   The IDMS is configured such that if the incoming Internet traffic volume exceeds 50% of the provisioned upstream Internet transit link

   capacity, the IDMS will request DDoS mitigation assistance from the
   upstream transit provider.

   The communication to trigger, manage, and finalize a DDoS mitigation
   between the enterprise IDMS and the transit provider is performed
   using DOTS. The enterprise IDMS implements a DOTS client while the
   transit provider implements a DOTS server.

   When the IDMS detects an inbound DDoS attack targeting the enterprise
   servers and applications, it immediately begins mitigating the
   attack.

   During the course of the attack, the inbound traffic volume exceeds
   the 50% threshold; the IDMS DOTS client signals the DOTS server on
   the upstream transit provider network to initiate DDoS mitigation.
   The DOTS server signals the DOTS client that it can service this
   request, and mitigation is initiated on the transit provider network.

   Over the course of the attack, the DOTS server on the transit
   provider network periodically signals the DOTS client on the
   enterprise IDMS in order to provide mitigation status information,
   statistics related to DDoS attack traffic mitigation, and related
   information. Once the DDoS attack has ended, the DOTS server signals
   the enterprise IDMS DOTS client that the attack has subsided.

   The enterprise IDMS then requests that DDoS mitigation services on
   the upstream transit provider network be terminated. The DOTS server
   on the transit provider network receives this request, communicates
   with the transit provider orchestration system controlling its DDoS
   mitigation system to terminate attack mitigation, and once the
   mitigation has ended, confirms the end of upstream DDoS mitigation
   service to the enterprise IDMS DOTS client.

3.1.2. Enterprise with on Cloud DDoS mitigation provider

   This use case details an enterprise that has a local DDoS detection
   and classification capability and may or may not have a mitigation
   capability. The enterprise is contracted with a cloud DDoS
   mitigation provider who can redirect (offramp) traffic away from the
   enterprise, provide scrubbing services and return clean traffic back
   to the enterprise (onramp) on an ad-hoc, on demand basis.

   The enterprise may, either by hard coding or on a case by case basis,
   determine thresholds at which a request for mitigation is triggered
   indicating to the cloud provider that traffic should be redirected
   and scrubbed.

The communication to trigger, manage, and finalize a DDoS mitigation
between the enterprise and the Cloud provider is performed using
DOTS. The enterprise implements a DOTS client while the Cloud
Provider implements a DOTS server.

The enterprise detection and classification systems encompass a DOTS
client and the cloud provider a DOTS server.

When an attack is detected an automated or manual DOTS mitigation
request will be generatd and sent to the cloud provider. The cloud
provider will assess the request for validity and if passed a
mitigation action may then be initiated. This action will usually
involve the offramp of all traffic destined to the target for further
scrutiny and filtering by the cloud provider. This should not only
result in an alleviation of pressure on the enterprise network but
also on its upstream provider and peers.

The cloud provider should signal via DOTS to the enterprise that a
mitigation request has been received and acted upon and should also
include a basic situational status of the attack. The cloud provider
may respond periodically with additional updates on the status to
enable the enterprise to make an informed decision on whether to
maintain or cancel the mitigation. An alternative approach would be
for the DOTS client mitigation request to include a time to live
(ttl) for the mitigation which may be extended by the client should
the attack still be ongoing as the ttl reaches expiration.

A variation of this use case may be that the enterprise is providing
a flow based monitoring and analysis service to customers whose
networks may be protected by any one of a number of 3rd party
providers. The enterprise in question may integrate with these 3rd
party providers using DOTS and signal accordingly when a customer is
attacked - the enterprise may then manage the life-cycle of the
attack on behalf of the enterprise.

3.2. Intra-domain Use Cases

3.2.1. Homenet DDoS protection by ISP

In this use case home networks or small businesses networks (SOHO),
subscribe with their upstream ISP a DDoS mitigation service.

Home networks run with limited bandwidth as well as limited routing
resources, while they are expected to provide services reachable from
the outside [RFC7368]. This makes such organizations some easy
targets to DDoS attacks. In addition, these DDoS attacks might even
not be noticed by the upstream ISP.

This scenario is considered as an intra-domain as ISPs have a specific relationship with these customers. The ISP is the connectivity provider, and in some cases, they even provides the CPE with a set of associated services. Moreover, in case of any connectivity issue the customer is likely to call the hotline. In order to improve the QoS of the connectivity as well as to automate the request for DDoS mitigation, ISP is likely to consider a standard mean for CPEs to notify when they are under a suspected DDoS. Such notification may be triggered automatically or manually. As the ISP and the customer share a common interest in mitigating the DDoS attack, this slightly differs from cases where a contract is negotiated with a third party, such as in the inter-domain use cases.

In most cases, CPEs are unlikely to diagnose whether an DDoS attack is ongoing or not and simply rely on the upstream equipment provided by the ISP for detection and potential mitigation.

The DDoS Mitigation service of the ISP may be hard coded or may be configured by the customer manually or automatically while the CPE is being connected to the Internet -- eventually the DHCP server may provide the DDoS Mitigation service via specific DHCP options.

The communication to trigger a DDoS mitigation between the home network and the ISP is performed using DOTS. The home network CPE implements a DOTS client while the ISP implements a DOTS server.

The DOTS Client on the CPE monitors the status of CPE's resource and link bandwidth usage. If something unusual happens based on preconfigured throughput or some heuristics methods, the DOTS Client sends a DOTS mitigation request to the ISP DOTS Server. Typically, a default configuration with no additional information associated to the DOTS mitigation request is expected. The ISP derives traffic to mitigate from the CPE IP address.

In some cases, the DOTS mitigation request contains options such as some IP addresses or prefixes that belongs to a whitelist or respectively to a blacklist. In this case, the white and black lists are not associated to some analysis performed by the CPE -- as the CPE is clearly not expected to analyze such attacks. Instead these are part of some configuration parameters. For example, in the case of small business, one may indicate specific legitimate IP addresses such as those used for VPNs, or third party services the company is likely to set a session. Similarly, the CPE may provides the IP addresses of the assets to be protected inside the network. Such options may include the IP address as well as a service description. Similarly to the previous blacklist and whitelist, such information are not derived from a traffic analysis performed by the CPE, but instead are more related to configuration parameters.

Upon receiving the DOTS mitigation request, the ISP acknowledges its
reception and confirms DDoS mitigation starts or not. Such feed back
is mostly to avoid retransmission of the request.

Note that the ISP is connected to multiple CPEs and as such the CPE
can potentially perform DDoS attack to the DOTS server. ISP may use
relays to absorbs the traffic. In addition, such attack may be
triggered by a large scale DDoS attack, which is expected to be
detected and mitigated by the upstream architecture.

ISP may activate mitigation for the traffic associated to the CPE
sending the alert or instead to the traffic associated to all CPE.
Such decisions are not part of DOTS, but instead depend on the
policies of the ISP network administrator.

It is unlikely the CPE will follow the status of the mitigation. The
ISP is only expected to inform the CPE the mitigation has been
stopped.

Upon receipt of such notification the CPE may re-activate the
monitoring jobs and thus is likely to provide some further DOTS
alert.

3.2.2. DDoS Orchestration

In this use case, one or multiple telemetry systems or monitoring
devices like a flow collector monitor a network -- typically an ISP
network. Upon detection of a DDoS attack, these telemetry systems
alert an orchestrator in charge of coordinating the various DDoS
mitigation systems within the domain. The telemetry systems may be
configured to provide some necessary or useful pieces of
informations, such as a preliminary analysis of the observation to
the orchestrator.

The orchestrator analyses the various information it receives from
specialized equipements, and elaborates one or multiple DDoS
mitigation strategies. In some case, a manual confirmation may also
be required to chose a proposed strategy or to start the DDoS
mitigation. The DDoS mitigation may consists in multiple steps such
as configuring the network, the various hardware or already
instantiated DDoS mitigation functions. In some cases, some specific
virtual DDoS mitigation functions need to be instantiated and
properly chained between each other. Eventually, the coordination of
the mitigation may involved external DDoS resources such as a transit
provider Section 3.1 or a cloud provider Section 3.1.2.

The communication to trigger a DDoS mitigation between the telemetry
and monitoring systems and the orchestrator is performed using DOTS.

The telemetry systems implements a DOTS client while the Orchestrator implements a DOTS server.

The communication between to select a DDoS strategy by a network administrator and the orchestrator is also performed using DOTS. The network administrator via its web interfaces implements a DOTS client while the Orchestrator implements a DOTS server.

The communication between the Orchestrator and the DDoS mitigation systems is performed using DOTS. The Orchestrator implements a DOTS client while the DDoS mitigation systems implement a DOTS server.

The configuration aspects of each DDoS mitigation systems, as well as the instantiations of DDoS mitigation functions or network configuration is not part of DOTS. Similarly the discovery of the available DDoS mitigation functions is not pat of DOTS.

The Telemetry or monitoring systems monitors each various traffic network and each performs their measurement tasks. They are configure so that when an event or some measurements reach a predefined level to report a DOTS mitigation request to the orchestrator. The DOTS mitigation request may be associated with some element such as specific reporting, or analysis.

Upon receipt of the DOTS mitigation request from the telemetry system, the orchestrator responds with an acknowledgement, to avoid retransmission of the request for mitigation. The status of the DDoS mitigation indicates the orchestrator is in an analysing phase. The orchestrator begins collecting various informations from various telemetry systems on the network in order to correlate the measurements and provide an analyse of the event. Eventually, the orchestrator may ask additional informations to the telemetry system that just sent the DOTS request, however, the collection of these information is performed outside DOTS.

The orchestrator may be configured to start a DDoS mitigation upon approval from a network administrator. The analysis from the orchestrator is reported to the network administrator via a web interface. If the network administrator decides to start the mitigation, she order through her web interface a DOTS client to send a request for DDoS mitigation. This request is expected to be associated with a context that identifies the DDoS mitigation selected.

Upon receiving the DOTS request for DDoS mitigation from the network administrator, the orchestrator orchestrates the DDoS mitigation according to the specified strategy. It status first indicates the DDoS mitigation is starting while not effective. In fact the

orchestrator is expected to proceed to a significant number of
configurations.

Orchestration of the DDoS mitigation systems works similarly as
described in Section 3.1 or Section 3.1.2. The orchestrator
indicates with its status the DDoS Mitigation is effective.

When the DDoS mitigation is finished on the DDoS mitigation systems,
the orchestrator indicates to the Telemetry systems as well as to the
network administrator the DDoS mitigation is finished.

4. Security Considerations

DOTS is at risk from three primary attacks: DOTS agent impersonation,
traffic injection, and signaling blocking. The DOTS protocol MUST be
designed for minimal data transfer to address the blocking risk.

Impersonation and traffic injection mitigation can be managed through
current secure communications best practices. DOTS is not subject to
anything new in this area. One consideration could be to minimize
the security technologies in use at any one time. The more needed,
the greater the risk of failures coming from assumptions on one
technology providing protection that it does not in the presence of
another technology.

Additional details of DOTS security requirements may be found in
[I-D.ietf-dots-requirements].

5. IANA Considerations

No IANA considerations exist for this document at this time.

6. Acknowledgments

TBD

7. References

7.1. Normative References

   [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>.

7.2. Informative References

   [APACHE]   "Apache mod_security", <https://www.modsecurity.org>.

   [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.

   [RFC6335] Cotton, M., Eggert, L., Touch, J., Westerlund, M., and S.
              Cheshire, "Internet Assigned Numbers Authority (IANA)
              Procedures for the Management of the Service Name and
              Transport Protocol Port Number Registry", BCP 165,
              RFC 6335, DOI 10.17487/RFC6335, August 2011,
              <http://www.rfc-editor.org/info/rfc6335>.

   [RFC7368] Chown, T., Ed., Arkko, J., Brandt, A., Troan, O., and J.
              Weil, "IPv6 Home Networking Architecture Principles",
              RFC 7368, DOI 10.17487/RFC7368, October 2014,
              <http://www.rfc-editor.org/info/rfc7368>.

   [RRL]   "BIND RRL", <https://deepthought.isc.org/article/AA-
              00994/0/Using-the-Response-Rate-Limiting-Feature-in-BIND-
              9.10.html>.

Authors' Addresses

   Roland Dobbins (editor)
   Arbor Networks
   30 Raffles Place
   Level 17 Chevron House
   Singapore 048622
   Singapore

   Email: rdobbins@arbor.net

   Stefan Fouant

   Email: stefan.fouant@copperriverit.com

Daniel Migault
Ericsson
8400 boulevard Decarie
Montreal, QC H4P 2N2
Canada

Phone: +1 514-452-2160
Email: daniel.migault@ericsson.com

Robert Moskowitz
HTT Consulting
Oak Park, MI 48237
USA

Email: rgm@labs.htt-consult.com

Nik Teague
Verisign Inc
12061 Bluemont Way
Reston, VA 20190
USA

Phone: +44 791 763 5384
Email: nteague@verisign.com

Liang Xia
Huawei
No. 101, Software Avenue, Yuhuatai District
Nanjing
China

Email: Frank.xialiang@huawei.com

Kaname Nishizuka
NTT Communications
GranPark 16F 3-4-1 Shibaura, Minato-ku
Tokyo 108-8118
Japan

Email: kaname@nttv6.jp

DOTS   R. Dobbins
Internet-Draft   Arbor Networks
Intended status: Informational   D. Migault
Expires: October 17, 2018   Ericsson
                                                        S. Fouant

                                                     R. Moskowitz
                                                   HTT Consulting
                                                        N. Teague
                                                         Verisign
                                                          L. Xia
                                                          Huawei
                                                     K. Nishizuka
                                               NTT Communications
                                                   April 15, 2018

                 Use cases for DDoS Open Threat Signaling
                        draft-ietf-dots-use-cases-12

Abstract

   The DDoS Open Threat Signaling (DOTS) effort is intended to provide
   protocols to facilitate interoperability across disparate DDoS
   mitigation solutions. This document presents use cases which
   describe the interactions expected between the DOTS components as
   well as DOTS messaging exchanges. These use cases are meant to
   identify the interacting DOTS components, how they collaborate and
   what are the typical information to be exchanged.

Copyright Notice

Table of Contents

1. Introduction

   At the time of writing, distributed denial-of-service (DDoS) attack
   mitigation solutions are largely based upon siloed, proprietary
   communications schemes with vendor lock-in as a side-effect. This
   can result in the configuration, provisioning, operation, and
   activation of these solutions being a highly manual and often time-
   consuming process. Additionally, coordinating multiple DDoS
   mitigation solutions simultaneously is fraught with both technical
   and process-related hurdles. This greatly increases operational
   complexity which, in turn, can degrade the efficacy of mitigations.

   The DDoS Open Threat Signaling (DOTS) effort is intended to specify
   protocols that facilitate interoperability between diverse DDoS
   mitigation solutions and ensure greater integration in term of
   mitigation requests and attack characterization patterns. As DDoS
   solutions are broadly heterogeneous among vendors, the primary goal

of DOTS is to provide high-level interaction amongst differing DDoS
solutions, such as initiating, terminating DDoS mitigation assistance
or requesting the status of a DDoS mitigation.

This document provides use cases to provide inputs for the design of
the DOTS protocol(s). The use cases are not exhaustive and future
use cases are expected to emerge as DOTS is adopted and evolves.

2. Terminology and Acronyms

   This document makes use of the same terminology and definitions as
   [I-D.ietf-dots-requirements]. In addition it uses the terms defined
   below:

   o DDoS Mitigation Service Provider: designates the administrative
      entity providing the DDoS Mitigation Service.

   o DDoS Mitigation Service: designates a DDoS service provided to a
      customer and which is scoped to mitigate DDoS attacks. Services
      usually involve Service Level Agreement (SLA) that have to be met.
      It is the responsibility of the service provider to instantiate
      the DDoS Mitigation System to meet these SLAs.

   o DDoS Mitigation System (DMS): A system that performs DDoS
      mitigation. The DDoS Mitigation System may be composed by a
      cluster of hardware and/or software resources, but could also
      involve an orchestrator that may take decisions such as
      outsourcing partial or more of the mitigation to another DDoS
      Mitigation System.

   o DDoS Mitigation: The action performed by the DDoS Mitigation
      System.

   o Internet Transit Provider (ITP): designates the entity that
      delivers the traffic to the network. It can be an Internet
      Service Provider (ISP), or an upstream entity delivering the
      traffic to the ISP.

3. Use Cases

3.1. Upstream DDoS Mitigation by an Upstream Internet Transit Provider

   This use case describes how an enterprise or a residential customer
   network may take advantage of a pre-existing relation with its
   Internet Transit Provider (ITP) in order to mitigate a DDoS attack
   targeting its network. To improve the clarity of our purpose, the
   targeted network will be designated as enterprise network, but the
   same scenario applies to any downstream network, including

residential network. As the ITP provides connectivity to the
enterprise network, it is already on the path of the inbound or
outbound traffic of the enterprise network and well aware of the
networking parameters associated to the enterprise network
connectivity. This eases both the configuration and the
instantiation of a DDoS Mitigation Service. This section considers
two kind of DDoS Mitigation Service between an enterprise network and
an ITP:

o The upstream ITP may instantiate a DDoS Mitigation System (DMS)
   upon receiving a request from the enterprise network. This
   typically corresponds to the case when the enterprise network is
   under attack.

o On the other hand, the ITP may identify an enterprise network as
   the source of an attack and send a mitigation request to the
   enterprise DMS to mitigate this at the source.

In the first scenario, as depicted in Figure 1, an enterprise network
with self-hosted Internet-facing properties such as Web servers,
authoritative DNS servers, and VoIP servers has a DMS deployed to
protect those servers and applications from DDoS attacks. In
addition to on-premise DDoS defense capability, enterprises have
contracted with their Internet transit provider for DDoS Mitigation
Services which threaten to overwhelm their WAN link(s) bandwidth.

```
  +-----------------+          +-----------------+
  | Entreprise      |          | Upstream        |
  | Network         |          | Internet Transit|
  |                 |          | Provider        |
  |     +--------+  |          |          DDoS Attack
  |     | DDoS   |  |  <================================
  |     | Target |  |  <================================
  |     +--------+  |          |   +-----------+  |
  |                 |  +-------->|   DDoS        |  |
  |                 |  |      |S |  Mitigation   |  |
  |                 |  |      |  |  System       |  |
  |                 |  |      |   +-----------+  |
  |                 |  |      |                  |
  |                 |  |      |                  |
  |                 |  |      |                  |
  |   +-----------+ |  |      |                  |
  |   | DDoS      |<---+      |                  |
  |   | Mitigation|C |        |                  |
  |   | System    |  |        |                  |
  |   +-----------+ |        |                  |
  +-----------------+        +-----------------+
```
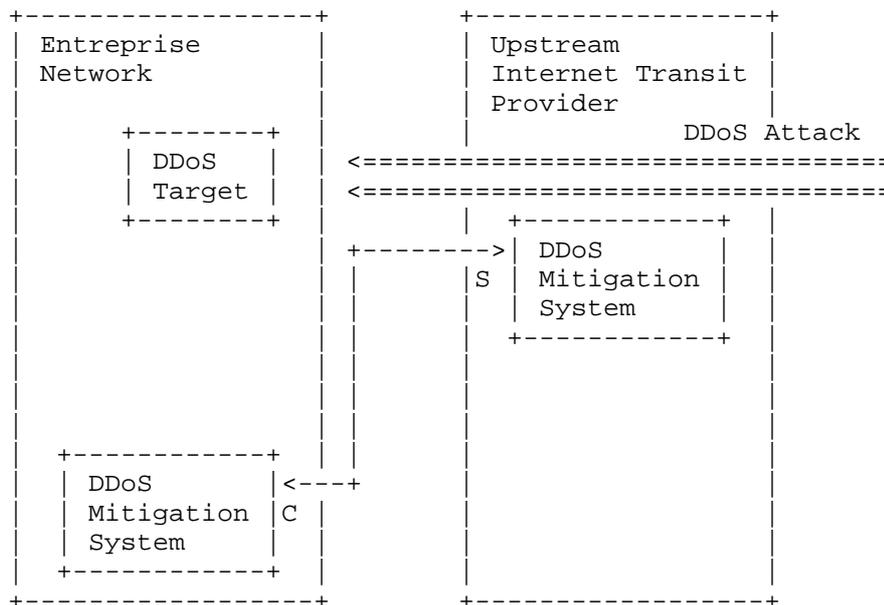
```
      * C is for DOTS client functionality
      * S is for DOTS server functionality
```

        Figure 1: Upstream Internet Transit Provider DDoS Mitigation

   The enterprise DMS is configured such that if the incoming Internet
   traffic volume exceeds 50% of the provisioned upstream Internet WAN
   link capacity, the DMS will request DDoS mitigation assistance from
   the upstream transit provider.

   The requests to trigger, manage, and finalize a DDoS Mitigation
   between the enterprise DMS and the ITP is performed using DOTS. The
   enterprise DMS implements a DOTS client while the ITP implements a
   DOTS server which is integrated with their DMS.

   When the enterprise DMS detects an inbound DDoS attack targeting its
   resources (e.g. servers, hosts or applications), it immediately
   begins a DDoS Mitigation.

   During the course of the attack, the inbound traffic volume exceeds
   the 50% threshold; the DMS DOTS client signals the DOTS server on the
   upstream ITP to initiate DDoS Mitigation. The DOTS server signals
   the DOTS client that it can serve this request, and mitigation is
   initiated on the ITP network by the ITP DMS.

Over the course of the attack, the DOTS server of the ITP
periodically informs the DOTS client on the enterprise DMS mitigation
status, statistics related to DDoS attack traffic mitigation, and
related information. Once the DDoS attack has ended, the DOTS server
signals the enterprise DMS DOTS client that the attack has subsided.

The enterprise DMS then requests the ITP to terminate the DDoS
Mitigation. The DOTS server on the ITP receives this request and
once the mitigation has ended, confirms the end of upstream DDoS
Mitigation to the enterprise DMS DOTS client.

The following is an overview of the DOTS communication model for this
use-case:

o (a) A DDoS attack is initiated against resources of a network
   organization which has deployed a DOTS-capable DMS - typically a
   DOTS client.

o (b) The DMS detects, classifies, and begins the DDoS Mitigation.

o (c) The DMS determines that its capacity and/or capability to
   mitigate the DDoS attack is insufficient, and sends via its DOTS
   client a DOTS DDoS Mitigation request to one or more DOTS servers
   residing on the upstream ITP.

o (d) The DOTS server which receive the DOTS Mitigation request
   determines that they have been configured to honor requests from
   the requesting DOTS client, and honored its DDoS Mitigation by
   orchestrating its DMS.

o (e) While the DDoS Mitigation is active, DOTS servers regularly
   transmit DOTS DDoS Mitigation status updates to the DOTS client.

o (f) The DOTS client transmits a DOTS DDoS Mitigation termination
   request to the DOTS server.

o (g) The DOTS server terminates DDoS Mitigation.

Note that communications between the enterprise DOTS client and the
upstream transit provider DOTS Server may take place in-band within
the main Internet WAN link between the enterprise and the ITP; out-
of-band via a separate, dedicated wireline network link utilized
solely for DOTS signaling; or out-of-band via some other form of
network connectivity such as a third-party wireless 4G network
connectivity.

Note also that a DOTS clients that sends a DOTS Mitigation request
may be also triggered by a network admin that manually confirms the

request to the upstream ITP, in which case the request may be sent
from an application such as a web browser or a dedicated mobile
application.

Note also that when the enterprise is multihomed and connected to
multiple upstream ITPs, each ITP is only able to provide a DDoS
Mitigation Service for the traffic it transits. As a result, the
enterprise network may require to coordinate the various DDoS
Mitigation Services associated to each link. More multi-homing
considerations are discussed in [I-D.boucadair-dots-multihoming].

The current scenario describes the case where the DDoS Target is in
the enterprise network while the secondary DMS is provided by the
upstream ITP. An alternate use case may consider the scenario where
the ITP informs the enterprise network it is involved into an ongoing
attack or that infected machines have been identified. In this case
the DOTS client and DOTS server roles are inverted. The DOTS client
is located in the ITP network and the DOTS server is hosted in the
enterprise network. The enterprise network is then responsible to
perform the DDoS Mitigation. In some case the DDoS Mitigation may be
delegated back to the upstream ITP, as described in this section.

3.2. DDoS Mitigation by a Third party DDoS Mitigation Service Provider

This use case differs from the previous use case described in
Section 3.1 in that the DDoS Mitigation Service is not provided by an
upstream ITP. In other words, as represented in Figure 2, the
traffic is not forwarded through the DDoS Mitigation Service Provider
by default. In order to steer the traffic to the DDoS Mitigation
Service Provider, some network configuration changes are required.
As such, this use case likely to match large enterprises or large
data centers, but not exclusively. Another typical scenario for this
use case is the relation between DDoS Mitigation Service Provider
forming an overlay of DMS. When an DDoS Mitigation Service Provider
mitigating a DDoS attack reaches it resources capacities, it may
chose to delegate the DDoS Mitigation to another DDoS Mitigation
Service Provider.

```
+-----------------+           +-----------------+
| Entreprise      |           | Upstream        |
| Network         |           | Internet Transit|
|                 |           | Provider        |
|     +--------+  |           |            DDoS Attack
|     | DDoS   |  | <================================
|     | Target |  | <================================
|     +--------+  |   +---------------------------+
|                 |   | |             |         | |
|                 |   | |   +-----------------+  | |
|                 |   | |                        |
|                 |   | |   +-----------------+  | |
|                 |   | |   | DDoS Mitigation |  | |
|                 |   | |   | Service Provider|  | |
|                 |   | |   |                 |  | |
|  +------------+ | |   |   +------------+  | | |
|  | DDoS       |<---+   |   | DDoS       |<----+ |
|  | Mitigation |C |     |   | Mitigation |S |    |
|  | System     | |      |   | System     | |    |
|  +------------+ |      |   +------------+ |    |
+-----------------+      +-----------------+
```
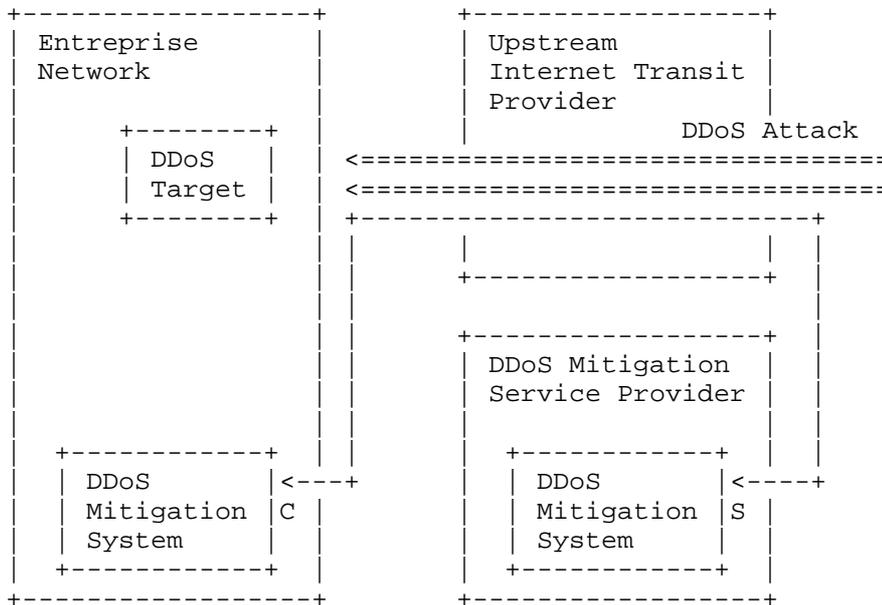
```
        * C is for DOTS client functionality
        * S is for DOTS server functionality
```

        Figure 2: DDoS Mitigation between an Enterprise Network and third
                party DDoS Mitigation Service Provider

   In this scenario, an Enterprise Network has entered into a pre-
   arranged DDoS mitigation assistance agreement with one or more other
   DDoS Mitigation Service Providers in order to ensure that sufficient
   DDoS mitigation capacity and/or capabilities may be activated in the
   event that a given DDoS attack threatens to overwhelm the ability of
   a given DMS to mitigate the attack on its own.

   The pre-arrangement typically includes the agreement on the
   mechanisms used to redirect the traffic to the DDoS Mitigation
   Service Provider, as well as the mechanism to to re-inject the
   traffic back to the Enterprise Network. Redirection to the DDoS
   Mitigation Service Provider typically involves BGP prefix
   announcement eventually combined with DNS redirection, while re-
   injection may be performed via tunneling mechanisms such as GRE for
   example. Of course, such mechanisms needs to be regularly tested and
   evaluated. These exact mechanisms used for traffic steering are out
   of scope.

```
        +-----------------+        +-----------------+
        | Entreprise      |        | Upstream        |
        | Network         |        | Internet Transit|
        |                 |        | Provider        |
        |                 |        |              DDoS Attack
        |      +--------+  |        |         |   ++====
        |      | DDoS   | |<----------------+  |   ||  ++=
        |      | Target | | Mitigated    |  |   ||  ||
        |      +--------+  |        |     |  |   ||  ||
        |                 |        |     |  |   ||  ||
        |                 |        +--------|--------+  ||  ||
        |                 |        |        |         ||  ||
        |                 |        +--------|--------+  ||  ||
        |                 |        | DDoS Mitigation |  ||  ||
        |                 |        | Service Provider|  ||  ||
        |                 |        |        |         ||  ||
        |  +-----------+  |        | +-----------+  |  ||  ||
        |  | DDoS      |<----------->| DDoS      |  |  ||  ||
        |  | mitigation|C |        |S | mitigation|<===++  ||
        |  | system    |  |        |  | system    |<======++
        |  +-----------+  |        | +-----------+  |
        +-----------------+        +-----------------+
```
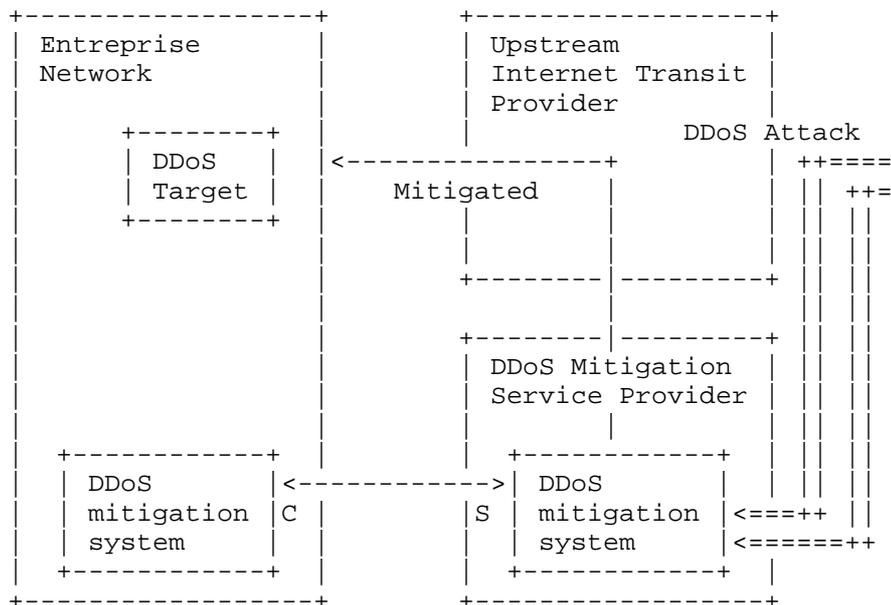
        * C is for DOTS client functionality
        * S is for DOTS server functionality

   Figure 3: Redirection to a DDoS Mitigation Service Provider

   When the Enterprise Network is under attack or at least is reaching
   its capacity or ability to mitigate a given DDoS attack traffic, the
   DOTS client sends a DOTS request to the DDoS Mitigation Service
   Provider to initiate network traffic diversion - as represented in
   Figure 3 - and DDoS mitigation activities. Ongoing attack and
   mitigation status messages may be passed between the Enterprise
   Network and the DDoS Mitigation Service Provider.

   Once the requesting Enterprise Network is confident that the DDoS
   attack has either ceased or has fallen to levels of traffic/
   complexity which they can handle on their own or that it has received
   a DOTS DDoS Mitigation termination request from a downstream
   Enterprise Network or DDoS Mitigation Service Provider, the
   requesting Enterprise Network DOTS client sends a DOTS DDoS
   Mitigation termination request to the DDoS Mitigation Service
   Provider.

3.3. DDoS Orchestration

   In this use case, one or more DDoS telemetry systems or monitoring
   devices monitor a network - typically an ISP network. Upon detection
   of a DDoS attack, these telemetry systems alert an orchestrator in
   charge of coordinating the various DMS within the domain. The
   telemetry systems may be configured to provide required information,
   such as a preliminary analysis of the observation to the
   orchestrator.

   The orchestrator analyses the various information it receives from
   specialized equipment, and elaborates one or multiple DDoS mitigation
   strategies. In some case, a manual confirmation may also be required
   to choose a proposed strategy or to initiate a DDoS Mitigation. The
   DDoS Mitigation may consist of multiple steps such as configuring the
   network, or updating already instantiated DDoS mitigation functions.
   In some cases, some specific DDoS mitigation functions must be
   instantiated and properly ordered. Eventually, the coordination of
   the mitigation may involve external DDoS resources such as a transit
   provider or a DDoS Mitigation Service Provider.

   The communications used to trigger a DDoS Mitigation between the
   telemetry and monitoring systems and the orchestrator is performed
   using DOTS. The telemetry systems implements a DOTS client while the
   orchestrator implements a DOTS server.

   The communication between a network administrator and the
   orchestrator is also performed using DOTS. The network administrator
   via its web interfaces implements a DOTS client, while the
   Orchestrator implements a DOTS server.

   The communication between the orchestrator and the DDoS mitigation
   systems is performed using DOTS. The orchestrator implements a DOTS
   Client while the DDoS mitigation systems implement a DOTS Server.

   The configuration aspects of each DDoS mitigation system, as well as
   the instantiations of DDoS mitigation functions or network
   configuration is not part of DOTS. Similarly, the discovery of
   available DDoS mitigation functions is not part of DOTS; and as such
   is out of scope.

```
            +----------+
            | network  |C
            | adminis  |<-+
            | trator   |  |
            +----------+  |
                          |                    (internal)
            +----------+  | S+--------------+   +-----------+
            |telemetry/|  +->|              |C S| DDoS      |+
            |monitoring|<--->|  Orchestrator|<--->| mitigation||
            |systems   |C  S |              |<-+  | systems   ||
            +----------+     +--------------+C |  +-----------+|
                                              |    +----------+
                                              |
                                              |  (external)
                                              |  +-----------+
                                              | S| DDoS      |
                                            +->|  mitigation|
                                              | | systems    |
                                              +-----------+
            * C is for DOTS client functionality
            * S is for DOTS server functionality
```
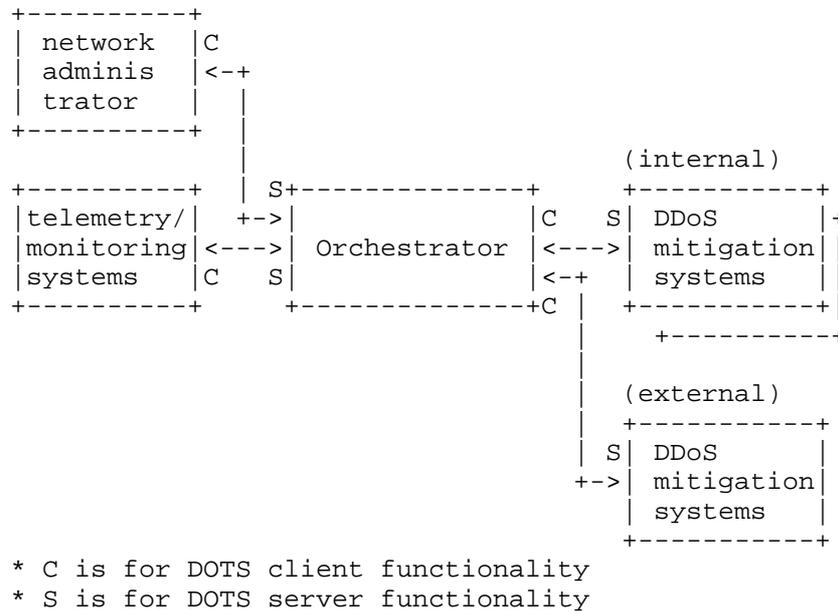
Figure 4: DDoS Orchestration

The telemetry systems monitor various network traffic and perform
some measurement tasks. These systems are configured so that when an
event or some measurement indicators reach a predefined level to
report a DOTS mitigation request to the orchestrator. The DOTS
mitigation request may be associated with some element such as
specific reporting.

Upon receipt of the DOTS mitigation request from the telemetry
system, the orchestrator responds with an acknowledgment, to avoid
retransmission of the request for mitigation. The status of the DDoS
mitigation indicates the orchestrator is in an analyzing phase. The
orchestrator begins collecting various information from various
telemetry systems in order to correlate the measurements and provide
an analysis of the event. Eventually, the orchestrator may ask
additional information to the telemetry system, however, the
collection of these information is out of scope.

The orchestrator may be configured to start a DDoS Mitigation upon
approval from a network administrator. The analysis from the
orchestrator is reported to the network administrator via a web
interface. If the network administrator decides to start the
mitigation, they order through her web interface a DOTS client to
send a request for DDoS mitigation. This request is expected to be

associated with a context that identifies the DDoS mitigation
selected.

Upon receiving the DOTS request for DDoS mitigation from the network
administrator, the orchestrator coordinates the DDoS mitigation
according to a specified strategy. Its status indicates the DDoS
mitigation is starting while not effective.

Orchestration of the DDoS mitigation systems works similarly as
described in Section 3.1. The orchestrator indicates with its status
whether the DDoS Mitigation is effective.

When the DDoS mitigation is finished on the DMS, the orchestrator
indicates to the telemetry systems as well as to the network
administrator the DDoS mitigation is finished.

4. Security Considerations

   DOTS is at risk from three primary attacks: DOTS agent impersonation,
   traffic injection, and signaling blocking. The DOTS protocol must be
   designed for minimal data transfer to address the blocking risk.

   Impersonation and traffic injection mitigation can be managed through
   current secure communications best practices. One consideration
   could be to minimize the security technologies in use at any one
   time. The more needed, the greater the risk of failures coming from
   assumptions on one technology providing protection that it does not
   in the presence of another technology.

   Additional details of DOTS security requirements can be found in
   [I-D.ietf-dots-requirements].

5. IANA Considerations

   No IANA considerations exist for this document at this time.

6. Acknowledgments

   The authors would like to thank among others Tirumaleswar Reddy;
   Andrew Mortensen; Mohamed Boucadaire; Artyom Gavrichenkov; Jon
   Shallow and the DOTS WG chairs, Roman D. Danyliw and Tobias Gondrom,
   for their valuable feedback.

7. Informative References

   [I-D.boucadair-dots-multihoming]
             Boucadair, M. and T. Reddy, "Multi-homing Deployment
             Considerations for Distributed-Denial-of-Service Open
             Threat Signaling (DOTS)", draft-boucadair-dots-
             multihoming-02 (work in progress), October 2017.

   [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-14 (work in
             progress), February 2018.

Authors' Addresses

   Roland Dobbins
   Arbor Networks
   Singapore

   EMail: rdobbins@arbor.net

   Daniel Migault
   Ericsson
   8275 Trans Canada Route
   Saint Laurent, QC   4S 0B6
   Canada

   EMail: daniel.migault@ericsson.com

   Stefan Fouant
   USA

   EMail: stefan.fouant@copperriverit.com

   Robert Moskowitz
   HTT Consulting
   Oak Park, MI   48237
   USA

   EMail: rgm@labs.htt-consult.com

   Nik Teague
   Verisign
   12061 Bluemont Way
   Reston, VA 20190

   EMail: nteague@verisign.com

   Liang Xia
   Huawei
   No. 101, Software Avenue, Yuhuatai District
   Nanjing
   China

   EMail: Frank.xialiang@huawei.com

   Kaname Nishizuka
   NTT Communications
   GranPark 16F 3-4-1 Shibaura, Minato-ku
   Tokyo 108-8118
   Japan

   EMail: kaname@nttv6.jp

DOTS   T. Reddy
Internet-Draft   Cisco
Intended status: Standards Track   M. Boucadair
Expires: September 11, 2017   Orange
                                                            K. Nishizuka
                                                      NTT Communications
                                                                 L. Xia
                                                                 Huawei
                                                               P. Patil
                                                                  Cisco
                                                            A. Mortensen
                                                    Arbor Networks, Inc.
                                                               N. Teague
                                                          Verisign, Inc.
                                                          March 10, 2017

        Distributed Denial-of-Service Open Threat Signaling (DOTS) Data Channel
                      draft-reddy-dots-data-channel-05

   Abstract

      The document specifies a Distributed Denial-of-Service Open Threat
      Signaling (DOTS) data channel used for bulk exchange of data not
      easily or appropriately communicated through the DOTS signal channel
      under attack conditions. This is a companion document to the DOTS
      signal channel specification.

Copyright Notice

Table of Contents

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 client, a router, a firewall, or an entire network.

    DDoS Open Threat Signaling (DOTS) defines two channels: signal and
    data channels [I-D.ietf-dots-architecture] (Figure 1). The DOTS
    signal channel used to convey that a network is under a DDOS attack
    to an upstream DOTS server so that appropriate mitigation actions are
    undertaken on the suspect traffic is further elaborated in
    [I-D.reddy-dots-signal-channel]. The DOTS data channel is used for
    infrequent bulk data exchange between DOTS agents in the aim to
    significantly augment attack response coordination.

```
   +--------------+                              +--------------+
   |              | <------- Signal Channel ------> |              |
   |  DOTS Client |                              |  DOTS Server |
   |              | <=======  Data Channel  ======> |              |
   +--------------+                              +--------------+
```
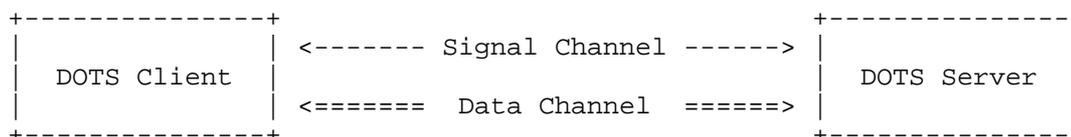
                        Figure 1: DOTS Channels

    Section 2 of [I-D.ietf-dots-architecture] identifies that the DOTS
    data channel is used to perform the tasks listed below:

    o Filter management, which enables a DOTS client to install or
       remove traffic filters, dropping or rate-limiting unwanted traffic
       and permitting white-listed traffic. Sample use cases for
       populating black- or white-list filtering rules are detailed
       hereafter:

       A. If a network resource (DOTS client) detects a potential DDoS
           attack from a set of IP addresses, the DOTS client informs its
           servicing router (DOTS gateway) of all suspect IP addresses
           that need to be blocked or black-listed for further
           investigation. The DOTS client could also specify a list of
           protocols and ports in the black-list rule. That DOTS gateway
           in-turn propagates the black-listed IP addresses to the DOTS
           server which will undertake appropriate action so that traffic
           from these IP addresses to the target network (specified by
           the DOTS client) is blocked.
       B. An enterprise network has partner sites from which only
           legitimate traffic arrives and the enterprise network wants to
           ensure that the traffic from these sites is not penalized

during DDOS attacks. The DOTS client uses DOTS data channel
to convey the white-listed IP addresses or prefixes of the
partner sites to its DOTS server. The DOTS server uses this
information to white-list flows from such IP addresses or
prefixes reaching the enterprise network.

o Creating identifiers, such as names or aliases, for resources for
   which mitigation may be requested:

   A. The DOTS client may submit to the DOTS server a collection of
       prefixes it wants to refer to by alias when requesting
       mitigation, to which the server would respond with a success
       status and the new prefix group alias, or an error status and
       message in the event the DOTS client's data channel request
       failed (see requirement OP-006 in [I-D.ietf-dots-requirements]
       and Section 2 in [I-D.ietf-dots-architecture]).

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].

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

   For simplicity, all of the examples in this document use "/restconf"
   as the discovered RESTCONF API root path. Many protocol header lines
   and message-body text within examples throughout the document are
   split into multiple lines for display purposes only. When a line
   ends with backslash ('\') as the last character, the line is wrapped
   for display purposes. It is to be considered to be joined to the
   next line by deleting the backslash, the following line break, and
   the leading whitespace of the next line.

3. DOTS Data Channel

   The DOTS data channel is intended to be used for bulk data exchanges
   between DOTS agents. Unlike the signal channel, which must operate
   nominally even when confronted with despite signal degradation due to
   packet loss, the data channel is not expected to be constructed to
   deal with attack conditions.

   As the primary function of the data channel is data exchange, a
   reliable transport is required in order for DOTS agents to detect
   data delivery success or failure. RESTCONF [RFC8040] over TLS
   [RFC5246] over TCP is used for DOTS data channel (Figure 2).
   RESTCONF uses HTTP methods to provide CRUD operations on a conceptual
   datastore containing YANG-defined data, which is compatible with a

server which implements NETCONF datastores. The HTTP POST, PUT,
PATCH, and DELETE methods are used to edit data resources represented
by DOTS data channel YANG data models. These basic edit operations
allow the DOTS data channel running configuration to be altered by a
DOTS client. DOTS data channel configuration data and state data can
be retrieved with the GET method. HTTP status codes are used to
report success or failure for RESTCONF operations. The DOTS client
will perform the root resource discovery procedure discussed in
Section 3.1 of [RFC8040] to determine the root of the RESTCONF API.
After discovering the RESTCONF API root, the DOTS client MUST use
this value as the initial part of the path in the request URI, in any
subsequent request to the DOTS server. The DOTS server can
optionally support retrieval of the YANG modules it supports
(Section 3.7 in [RFC8040]), for example, DOTS client can use RESTCONF
to retreive the company proprietary YANG model supported by the DOTS
server.

Note: This document uses RESTCONF, a protocol based on HTTP
[RFC7230], for configuring data defined in YANG version 1 [RFC6020]
or YANG version 1.1 [RFC7950], using the datastore concepts defined
in the Network Configuration Protocol (NETCONF) [RFC6241]. RESTCONF
combines the simplicity of the HTTP protocol with the predictability
and automation potential of a schema-driven API. RESTCONF offers a
simple subset of NETCONF functionality and provides a simplified
interface using REST-like API which addresses the needs of the DOTS
data channel and hence an optimal choice.

```
                      +--------------+
                      |    DOTS      |
                      +--------------+
                      |  RESTCONF    |
                      +--------------+
                      |    TLS       |
                      +--------------+
                      |    TCP       |
                      +--------------+
                      |    IP        |
                      +--------------+
```
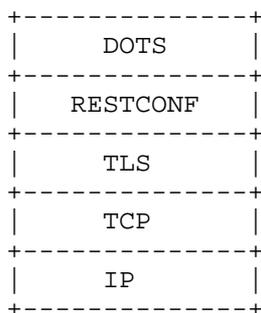
      Figure 2: Abstract Layering of DOTS data channel over RESTCONF over
                                    TLS

JavaScript Object Notation (JSON) [RFC7159] payload is used to
propogate data channel specific payload messages that convey request
parameters and response information such as errors. This
specification uses the encoding rules defined in [RFC7951] for
representing DOTS data channel configuration data defined using YANG
(Section 3.1) as JSON text.

   A DOTS client registers itself to its DOTS server(s) in order to set
   up DOTS data channel related configuration data on the DOTS server
   and receive state data (i.e., non-configuration data) from the DOTS
   server. A single DOTS data channel between DOTS agents can be used
   to exchange multiple requests and multiple responses. To reduce DOTS
   client and DOTS server workload, DOTS client SHOULD re-use the TLS
   session. While the communication to the DOTS server is quiescent,
   the DOTS client MAY probe the server to ensure it has maintained
   cryptographic state. Such probes can also keep alive firewall or NAT
   bindings. A TLS heartbeat [RFC6520] verifies the DOTS server still
   has TLS state by returning a TLS message.

3.1. DOTS Data Channel YANG Model

3.1.1. Identifier Model structure

   This document defines a YANG [RFC6020] data model for creating
   identifers, such as names or aliases, for resources for which
   mitigation may be requested. Such identifiers may then be used in
   subsequent DOTS signal channel exchanges to refer more efficiently to
   the resources under attack.

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

   module: ietf-dots-data-channel-identifier
       +--rw identifier
          +--rw alias* [alias-name]
             +--rw alias-name   string
             +--rw ip*   inet:ip-address
             +--rw prefix*   inet:ip-prefix
             +--rw port-range* [lower-port upper-port]
             | +--rw lower-port   inet:port-number
             | +--rw upper-port   inet:port-number
             +--rw traffic-protocol*   uint8
             +--rw FQDN*   inet:domain-name
             +--rw URI*   inet:uri
             +--rw E.164*   string

3.1.2. Identifier Model

<CODE BEGINS> file "ietf-dots-data-channel-identifier@2016-11-28.yang"

module ietf-dots-data-channel-identifier {
     namespace "urn:ietf:params:xml:ns:yang:ietf-dots-data-channel-identifier";
     prefix "alias";
     import ietf-inet-types {
         prefix "inet";

```
 }
      organization "Cisco Systems, Inc.";
      contact "Tirumaleswar Reddy <tireddy@cisco.com>";

      description
        "This module contains YANG definition for
         configuring identifiers for resources using DOTS data channel";

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

      container identifier {
          description "top level container for identifiers";
              list alias {
                  key alias-name;
                  description "list of identifiers";
                  leaf alias-name {
                      type string;
                      description "alias name";
                  }
                  leaf-list ip {
                      type inet:ip-address;
                      description "IP address";
                  }
                  leaf-list prefix {
                      type inet:ip-prefix;
                      description "prefix";
                  }
                  list 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 traffic-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 E.164 {
                            type string;
                            description "E.164 number";
                        }
                    }
            }
        }
    <CODE ENDS>
```

3.1.3. Filter Model and structure

   This document uses the Access Control List (ACL) YANG data model
   [I-D.ietf-netmod-acl-model] for the configuration of filtering rules.
   ACL is explained in Section 1 of [I-D.ietf-netmod-acl-model].

   Examples of such configuration include:

   o Black-list management, which enables a DOTS client to inform the
      DOTS server about sources from which traffic should be suppressed.
   o White-list management, which enables a DOTS client to inform the
      DOTS server about sources from which traffic should always be
      accepted.
   o Filter management, which enables a DOTS client to install or
      remove traffic filters, dropping or rate-limiting unwanted traffic
      and permitting white-listed traffic.

3.2. Identifiers

3.2.1. Create Identifiers

   A POST request is used to create identifiers, such as names or
   aliases, for resources for which a mitigation may be requested. Such
   identifiers may then be used in subsequent DOTS signal channel
   exchanges to refer more efficiently to the resources under attack
   (Figure 3).

```
 POST /restconf/data/ietf-dots-data-channel-identifier HTTP/1.1
 Host: {host}:{port}
 Content-Format: "application/yang.api+json"
 {
  "ietf-dots-data-channel-identifier:identifier": {
    "alias": [
      {
        "alias-name": "string",
        "ip": [
          "string"
        ],
        "prefix": [
          "string"
        ],
        "port-range": [
          {
            "lower-port": integer,
            "upper-port": integer
          }
        ],
        "traffic-protocol": [
          integer
        ],
        "FQDN": [
          "string"
        ],
        "URI": [
          "string"
        ],
        "E.164": [
          "string"
        ]
      }
    ]
  }
 }
```

                    Figure 3: POST to create identifiers

   The header parameters are described below:

   alias-name: Name of the alias. This is a mandatory attribute.
   traffic-protocol: Internet Protocol numbers. This is an optional
      attribute.
   port-range: The port range, lower-port for lower port number and
      upper-port for upper port number. For TCP, UDP, SCTP, or DCCP:

      the range of ports (e.g., 80 to 8080). This is an optional
      attribute.
   ip: IP addresses are separated by commas. This is an optional
      attribute.
   prefix: Prefixes are separated by commas. This is an optional
      attribute.
   FQDN: Fully Qualified Domain Name, 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: Uniform Resource Identifier (URI). This is an optional
      attribute.
   E.164:   E.164 number. This is an optional attribute.

   In the POST request at least one of the attributes ip or prefix or
   FQDN or URI MUST be present. DOTS agents can safely ignore Vendor-
   Specific parameters they don't understand.

   Figure 4 shows a POST request to create alias called "https1" for
   HTTP(S) servers with IP addresses 2002:db8:6401::1 and
   2002:db8:6401::2 listening on port 443.

```
POST /restconf/data/ietf-dots-data-channel-identifier HTTP/1.1
Host: www.example.com
Content-Format: "application/yang.api+json"
{
   "ietf-dots-data-channel-identifier:identifier": {
     "alias": [
       {
         "alias-name": "Server1",
         "traffic-protocol": [
          6
         ],
         "ip": [
           "2002:db8:6401::1",
           "2002:db8:6401::2"
         ],
         "port-range": [
           {
             "lower-port": 443
           }
         ]
       }
     ]
   }
}
```

                   Figure 4: POST to create identifiers

The DOTS server indicates the result of processing the POST request
using HTTP response codes. HTTP 2xx codes are success, HTTP 4xx
codes are some sort of invalid requests and 5xx codes are returned if
the DOTS server has erred or it is incapable of accepting the alias.
Response code 201 (Created) will be returned in the response if the
DOTS server has accepted the alias. If the request is missing one or
more mandatory attributes then 400 (Bad Request) will be returned in
the response or if the request contains invalid or unknown parameters
then 400 (Invalid query) will be returned in the response. The HTTP
response will include the JSON body received in the request.

The DOTS client can use the PUT request (Section 4.5 in [RFC8040]) to
create or modify the aliases in the DOTS server.

3.2.2. Delete Identifiers

A DELETE request is used to delete identifiers maintained by a DOTS
server (Figure 5).

```
DELETE /restconf/data/ietf-dots-data-channel-identifier:identifier\
       /alias=Server1 HTTP/1.1
Host: {host}:{port}
```

Figure 5: DELETE identifier

In RESTCONF, URI-encoded path expressions are used. A RESTCONF data
resource identifier is encoded from left to right, starting with the
top-level data node, according to the "api-path" rule defined in
Section 3.5.3.1 of [RFC8040]. The data node in the above path
expression is a YANG list node and MUST be encoded according to the
rules defined in Section 3.5.1 of [RFC8040].

If the DOTS server does not find the alias name conveyed in the
DELETE request in its configuration data, then it responds with a 404
(Not Found) error response code. The DOTS server successfully
acknowledges a DOTS client's request to remove the identifier using
204 (No Content) in the response.

3.2.3. Retrieving Installed Identifiers

A GET request is used to retrieve the set of installed identifiers
from a DOTS server (Section 3.3.1 in [RFC8040]). Figure 6 shows how
to retrieve all the identifiers that were instantiated by the DOTS
client. The content parameter and its permitted values are defined
in Section 4.8.1 of [RFC8040].

```
GET /restconf/data/ietf-dots-data-channel-identifier:identifier?\
    content=config HTTP/1.1
Host: {host}:{port}
Accept: application/yang-data+json
```

Figure 6: GET to retrieve all the installed identifiers

Figure 7 shows response for all identifiers on the DOTS server.

```
   {
    "ietf-dots-data-channel-identifier:identifier": [
      {
        "alias": [
          {
            "alias-name": "Server1",
            "traffic-protocol": [
              6
            ],
            "ip": [
              "2002:db8:6401::1",
              "2002:db8:6401::2"
            ],
            "port-range": [
              {
                "lower-port": 443
              }
            ]
          }
        ]
      },
      {
        "alias": [
          {
            "alias-name": "Server2",
            "traffic-protocol": [
              6
            ],
            "ip": [
              "2002:db8:6401::10",
              "2002:db8:6401::20"
            ],
            "port-range": [
              {
                "lower-port": 80
              }
            ]
          }
        ]
      }
    ]
   }
```

                      Figure 7: Response body

   If the DOTS server does not find the alias name conveyed in the GET
   request in its configuration data, then it responds with a 404 (Not
   Found) error response code.

3.3. Filtering Rules

   The DOTS server either receives the filtering rules directly from the
   DOTS client or via the DOTS gateway. If the DOTS client signals the
   filtering rules via the DOTS gateway then the DOTS gateway validates
   if the DOTS client is authorized to signal the filtering rules and if
   the client is authorized propagates the rules to the DOTS server.
   Likewise, the DOTS server validates if the DOTS gateway is authorized
   to signal the filtering rules. To create or purge filters, the DOTS
   client sends HTTP requests to the DOTS gateway. The DOTS gateway
   validates the rules in the requests and proxies the requests
   containing the filtering rules to a DOTS server. When the DOTS
   gateway receives the associated HTTP response from the DOTS server,
   it propagates the response back to the DOTS client.

   The following APIs define means for a DOTS client to configure
   filtering rules on a DOTS server.

3.3.1. Install Filtering Rules

   A POST request is used to push filtering rules to a DOTS server.
   Figure 8 shows a POST request example to block traffic from
   10.10.10.1/24, destined to 11.11.11.1/24. The ACL JSON configuration
   for the filtering rule is generated using the ACL YANG data model
   defined in [I-D.ietf-netmod-acl-model] and the ACL configuration XML
   for the filtering rule is specified in Section 4.3 of
   [I-D.ietf-netmod-acl-model]. This specification updates the ACL YANG
   data model defined in [I-D.ietf-netmod-acl-model] to support rate-
   limit action.

```
POST /restconf/data/ietf-access-control-list HTTP/1.1
Host: www.example.com
Content-Format: "application/yang.api+json"
{
 "ietf-access-control-list:access-lists": {
    "acl": [
        {
            "acl-name": "sample-ipv4-acl",
            "acl-type": "ipv4",
            "access-list-entries": {
                "ace": [
                    {
                        "rule-name": "rule1",
                        "matches": {
                            "source-ipv4-network": "10.10.10.1/24",
                            "destination-ipv4-network": "11.11.11.1/24"
                        },
                        "actions": {
                            "deny": [null]
                        }
                    }
                ]
            }
        }
    ]
 }
}
```

Figure 8: POST to install filterng rules

The header parameters defined in [I-D.ietf-netmod-acl-model] are
discussed below:

acl-name: The name of access-list. This is a mandatory attribute.
acl-type: Indicates the primary intended type of match criteria
   (e.g. IPv4, IPv6). This is a mandatory attribute.
protocol: Internet Protocol numbers. This is an optional
   attribute.
source-ipv4-network: The source IPv4 prefix. This is an optional
   attribute.
destination-ipv4-network: The destination IPv4 prefix. This is an
   optional attribute.
actions: "deny" or "permit" or "rate-limit". "permit" action is
   used to white-list traffic. "deny" action is used to black-list
   traffic. "rate-limit" action is used to rate-limit traffic, the
   allowed traffic rate is represented in bytes per second indicated
   in IEEE floating point format [IEEE.754.1985]. If actions

attribute is not specified in the request then the default action
is "deny". This is an optional attribute.

The DOTS server indicates the result of processing the POST request
using HTTP response codes. HTTP 2xx codes are success, HTTP 4xx
codes are some sort of invalid requests and 5xx codes are returned if
the DOTS server has erred or it is incapable of configuring the
filtering rules. Response code 201 (Created) will be returned in the
response if the DOTS server has accepted the filtering rules. If the
request is missing one or more mandatory attributes then 400 (Bad
Request) will be returned in the response or if the request contains
invalid or unknown parameters then 400 (Invalid query) will be
returned in the response.

The DOTS client can use the PUT request to create or modify the
filtering rules in the DOTS server.

3.3.2. Remove Filtering Rules

A DELETE request is used to delete filtering rules from a DOTS server
(Figure 9).

```
DELETE /restconf/data/ietf-access-control-list:access-lists/acl-name\
      =sample-ipv4-acl&acl-type=ipv4 HTTP/1.1
Host: {host}:{port}
```

Figure 9: DELETE to remove the filtering rules

If the DOTS server does not find the access list name and access list
type conveyed in the DELETE request in its configuration data, then
it responds with a 404 (Not Found) error response code. The DOTS
server successfully acknowledges a DOTS client's request to withdraw
the filtering rules using 204 (No Content) response code, and removes
the filtering rules as soon as possible.

3.3.3. Retrieving Installed Filtering Rules

The DOTS client periodically queries the DOTS server to check the
counters for installed filtering rules. A GET request is used to
retrieve filtering rules from a DOTS server. Figure 10 shows how to
retrieve all the filtering rules programmed by the DOTS client and
the number of matches for the installed filtering rules.

```
  GET /restconf/data/ietf-access-control-list:access-lists?content=all HTTP/1.1
  Host: {host}:{port}
  Accept: application/yang-data+json
```

Figure 10: GET to retrieve the configuration data and state data for
                          the filtering rules

If the DOTS server does not find the access list name and access list
type conveyed in the GET request in its configuration data, then it
responds with a 404 (Not Found) error response code.

4. IANA Considerations

   This specification registers new parameters for the DOTS data channel
   and establishes registries for mappings to JSON attributes.

4.1. DOTS Data Channel JSON Attribute Mappings Registry

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

4.2. Registration Template

```
   JSON Attribute:
      JSON attribute name.
   Description:
      Brief description of the attribute.
   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.
```

4.3. Initial Registry Contents

   o JSON Attribute: "alias-name"
   o Description: Name of alias.
   o Change Controller: IESG
   o Specification Document(s): this document

   o JSON Attribute: "traffic-protocol"
   o Description: Internet protocol numbers.
   o Change Controller: IESG

   o Specification Document(s): this document

   o JSON Attribute: "port-range"
   o Description: The port range, lower-port for lower port number and
      upper-port for upper port number. For TCP, UDP, SCTP, or DCCP:
      the range of ports (e.g., 80 to 8080).
   o Change Controller: IESG
   o Specification Document(s): this document

   o JSON Attribute: "lower-port"
   o Description: Lower port number for port range.
   o Change Controller: IESG
   o Specification Document(s): this document

   o JSON Attribute: "upper-port"
   o Description: Upper port number for port range.
   o Change Controller: IESG
   o Specification Document(s): this document

   o JSON Attribute: "ip"
   o Description: IP address.
   o Change Controller: IESG
   o Specification Document(s): this document

   o JSON Attribute: "prefix"
   o Description: IP prefix
   o Change Controller: IESG
   o Specification Document(s): this document

   o JSON Attribute: "FQDN"
   o Description: Fully Qualified Domain Name, 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.
   o Change Controller: IESG
   o Specification Document(s): this document

   o JSON Attribute: "URI"
   o Description: Uniform Resource Identifier (URI).
   o Change Controller: IESG
   o Specification Document(s): this document

   o JSON Attribute: "E.164"
   o Description: E.164 number.
   o Change Controller: IESG
   o Specification Document(s): this document

5. Contributors

    The following individuals have contributed to this document:

    Dan Wing Email: dwing-ietf@fuggles.com

6. Security Considerations

    Authenticated encryption MUST be used for data confidentiality and
    message integrity. TLS based on client certificate MUST be used for
    mutual authentication. The interaction between the DOTS agents
    requires Transport Layer Security (TLS) with a cipher suite offering
    confidentiality protection and the guidance given in [RFC7525] MUST
    be followed to avoid attacks on TLS.

    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.

7. Acknowledgements

    Thanks to Christian Jacquenet, Roland Dobbins, Andrew Mortensen,
    Roman Danyliw, Ehud Doron and Gilbert Clark for the discussion and
    comments.

8. References

8.1. Normative References

   [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-netmod-acl-model]
              Bogdanovic, D., Koushik, K., Huang, L., and D. Blair,
              "Network Access Control List (ACL) YANG Data Model",
              draft-ietf-netmod-acl-model-09 (work in progress), October
              2016.

   [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>.

   [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>.

   [RFC7951] Lhotka, L., "JSON Encoding of Data Modeled with YANG",
              RFC 7951, DOI 10.17487/RFC7951, August 2016,
              <http://www.rfc-editor.org/info/rfc7951>.

   [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF
              Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017,
              <http://www.rfc-editor.org/info/rfc8040>.

8.2. Informative References

   [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-03 (work in
            progress), October 2016.

   [I-D.reddy-dots-signal-channel]
            Reddy, T., Boucadair, M., and P. Patil, "Distributed
            Denial-of-Service Open Threat Signaling (DOTS) Signal
            Channel", draft-reddy-dots-signal-channel-09 (work in
            progress), March 2017.

   [IEEE.754.1985]
            Institute of Electrical and Electronics Engineers,
            "Standard for Binary Floating-Point Arithmetic", August
            1985.

   [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>.

   [RFC6520] Seggelmann, R., Tuexen, M., and M. Williams, "Transport
            Layer Security (TLS) and Datagram Transport Layer Security
            (DTLS) Heartbeat Extension", RFC 6520,
            DOI 10.17487/RFC6520, February 2012,
            <http://www.rfc-editor.org/info/rfc6520>.

   [RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data
            Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March
            2014, <http://www.rfc-editor.org/info/rfc7159>.

Authors' Addresses

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

   Email: tireddy@cisco.com

Mohamed Boucadair
Orange
Rennes 35000
France

Email: mohamed.boucadair@orange.com

Kaname Nishizuka
NTT Communications
GranPark 16F 3-4-1 Shibaura, Minato-ku
Tokyo 108-8118
Japan

Email: kaname@nttv6.jp

Liang Xia
Huawei
101 Software Avenue, Yuhuatai District
Nanjing, Jiangsu 210012
China

Email: frank.xialiang@huawei.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

        Distributed Denial-of-Service Open Threat Signaling (DOTS) Signal
                                   Channel
                      draft-reddy-dots-signal-channel-09

   Abstract

      This document specifies a mechanism that a DOTS client can use to
      signal that a network is under a Distributed Denial-of-Service (DDoS)
      attack to an upstream DOTS server so that appropriate mitigation
      actions are undertaken (including, blackhole, drop, rate-limit, or
      add to watch list) on the suspect traffic. The document specifies
      the DOTS signal channel including Happy Eyeballs considerations. The
      specification of the DOTS data channel is elaborated in a companion
      document.

   Status of This Memo

   Copyright Notice

Table of Contents

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 an network administrators
   to determine the causes of an attack, but instead just realize that
   certain resources seem to be under attack. This document, which
   adheres to the DOTS architecture [I-D.ietf-dots-architecture],
   proposes that, in such cases, the DOTS client just inform its DOTS
   server(s) that the network is under a potential attack and that the
   mitigator monitors traffic to the network to mitigate any possible
   attacks. This cooperation between DOTS agents contributes to ensure
   a highly automated network that is also robust, reliable and secure.

   Protocol requirements for DOTS signal channel are obtained from DOTS
   requirements [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].

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 and the firewall 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.

```
   Network
   Resource            CPE router          Access network    _____
 +-----------+      +--------------+      +-------------+    /          \
 |           |_____|              |_____|             |___|  Internet  |
 |DOTS client|      | DOTS gateway |      | DOTS server |   |            |
 |           |      |              |      |             |   |            |
 +-----------+      +--------------+      +-------------+    _____/
```
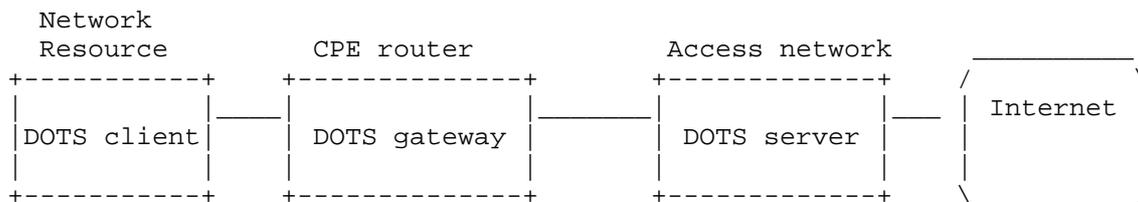
Figure 1

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

```
    Network                                              DDoS mitigation
    Resource              CPE router         _____      service
   +----------+        +-------------+      /          \  +-------------+
   |          |____    |             |_____|            |__|             |
   |DOTS client|       |DOTS gateway |     |  Internet  |  | DOTS server |
   |          |        |             |     |            |  |             |
   +----------+        +-------------+      _____/   +-------------+
```
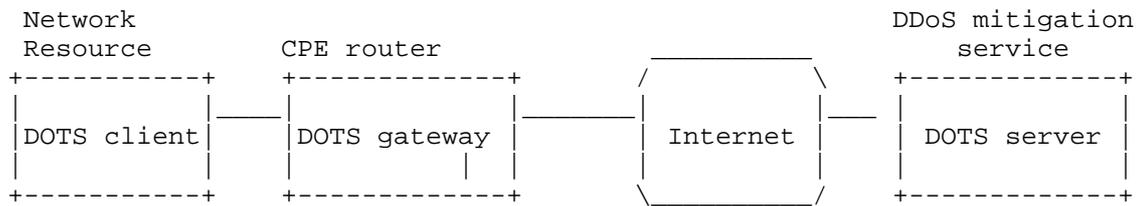
                              Figure 2

   In typical deployments, the DOTS client belongs to a different
   administrative domain than the DOTS server. For example, the DOTS
   client is a web server serving content 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 its 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).

```
DOTS client                                                   DOTS server
   |                                                              |
   |--DTLS ClientHello, IPv6 ---->X                               |
   |--TCP SYN, IPv6-------------->X                               |
   |--DTLS ClientHello, IPv4 ---->X                               |
   |--TCP SYN, IPv4--------------------------------------------->|
   |--DTLS ClientHello, IPv6 ---->X                               |
   |--TCP SYN, IPv6-------------->X                               |
   |<-TCP SYNACK------------------------------------------------|
   |--DTLS ClientHello, IPv4 ---->X                               |
   |--TCP ACK--------------------------------------------------->|
   |<------------Establish TLS Session------------------------->|
   |---------------DOTS signal---------------------------------->|
   |                                                              |
```
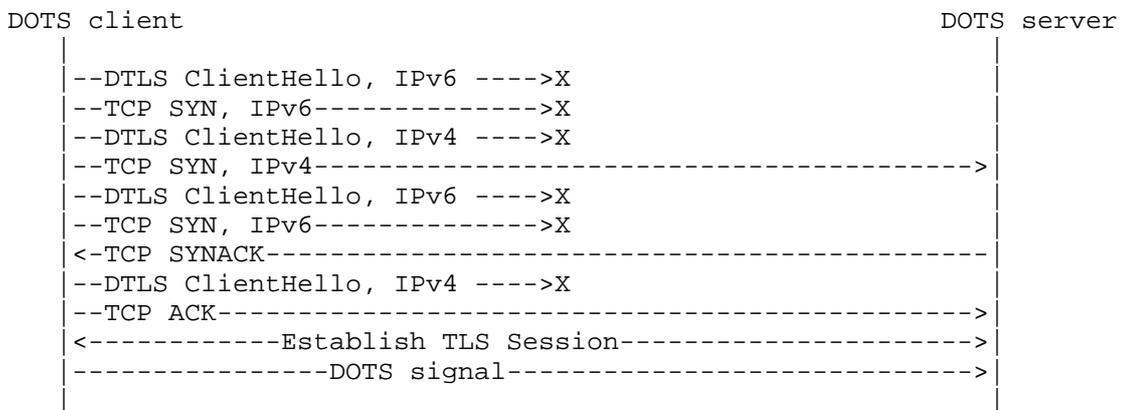
                       Figure 3: Happy Eyeballs

   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 IPv6
   path and UDP over IPv6 and IPv4 is retried until the DOTS client
   gives up.

5. DOTS Signal Channel

5.1. Overview

   Constrained Application Protocol (CoAP) [RFC7252] is used for DOTS
   signal channel (Figure 4). COAP was designed according to the REST
   architecture, and thus exhibits functionality similar to that of
   HTTP, it is quite straightforward to map from CoAP to HTTP and from
   HTTP to CoAP. CoAP has been defined to make use of both DTLS over

UDP and TLS over TCP [I-D.ietf-core-coap-tcp-tls]. The advantages of
COAP are: (1) Like HTTP, CoAP is based on the successful REST model,
(2) CoAP is designed to use minimal resources, (3) CoAP integrates
with JSON, CBOR or any other data format, (4) asynchronous message
exchanges, (5) includes a congestion control mechanism (6) allows
configuration of message transmission parameters specific to the
application environment (including dynamically adjusted values, see
Section 4.8.1 in [RFC7252]) etc.

```
                    +--------------+
                    |    DOTS      |
                    +--------------+
                    |    CoAP      |
                    +--------------+
                    | TLS | DTLS   |
                    +--------------+
                    | TCP |  UDP   |
                    +--------------+
                    |     IP       |
                    +--------------+
```
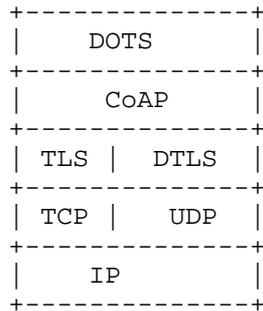
         Figure 4: Abstract Layering of DOTS signal channel over CoAP over
                                 (D)TLS

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.

Concise Binary Object Representation (CBOR) [RFC7049] 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 DOTS signal channel
configuration data defined using YANG (Section 5.2) as CBOR data.

5.2. DOTS Signal YANG Model

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 E.164*             string
             +--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 <tireddy@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 E.164 {
                     type string;
                     description "E.164 number";
                }
                leaf-list alias {
                    type string;
                    description "alias name";
                }
                leaf lifetime {
                    type int32;
```

```
                       description "lifetime";
                }
            }
        }
    }
<CODE ENDS>
```

5.2.3. Session Configuration Model structure

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

```
   module: ietf-dots-signal-config
       +--rw signal-config
          +--rw policy-id?           int32
          +--rw heartbeat-interval?  int16
          +--rw max-retransmit?      int16
          +--rw ack-timeout?         int16
          +--rw ack-random-factor?   decimal64
```

5.2.4. Session Configuration Model

```
<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 <tireddy@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-interval {
            type int16;
            description "heartbeat interval";
        }
        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 APIs define the means to convey a DOTS signal from a
   DOTS client to a DOTS server:

   PUT requests: are used to convey the DOTS signal from a DOTS client
      to a DOTS server over the signal channel, possibly traversing a
      DOTS gateway, indicating the DOTS client's need for mitigation, as
      well as the scope of any requested mitigation (Section 5.3.1).
      DOTS gateway act as a CoAP-to-CoAP Proxy (explained in [RFC7252]).
      PUT requests are also used by the DOTS client to convey mitigation
      efficacy updates to the DOTS server (Section 5.3.4).
   DELETE requests: are used by the DOTS client to withdraw the request
      for mitigation from the DOTS server (Section 5.3.2).
   GET requests: are used by the DOTS client to retrieve the DOTS
      signal(s) it had conveyed to the DOTS server (Section 5.3.3).

   Reliability is provided to the requests and responses by marking them
   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]).

   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]).

   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).

5.3.1. Convey DOTS Signals

When suffering an attack and desiring DoS/DDoS mitigation, a DOTS
signal is sent by the DOTS client to the DOTS server. A PUT request
is used to convey a DOTS signal 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 any mitigator feedback
to the requesting DOTS client. The PUT request and response are
marked as Non-confirmable messages.

```
     Header: PUT (Code=0.03)
     Uri-Host: "host"
     Uri-Path: ".well-known"
     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"
              ],
              "E.164": [
                "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. IP addresses are
   separated by commas. This is an optional attribute.
target-prefix: A list of prefixes under attack. Prefixes are
   separated by commas. 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.
E.164:   A list of E.164 numbers. 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. 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. The mitigation request with
   higher numeric policy-id value has higher precedence (and thus will
   match before) than the mitigation request with lower numeric policy-
   id value.

   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: ".well-known"
   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). The DELETE request and response are marked as
Confirmable messages.

```
Header: DELETE (Code=0.04)
Uri-Host: "host"
Uri-Path: ".well-known"
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.

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 GET request is marked as Non-confirmable message.
   The 'c' (content) parameter and its permitted values defined in
   [I-D.ietf-core-comi] can be used to retreive 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: ".well-known"
      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: ".well-known"
      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.                                         |
\-------------------+-----------------------------------------------------/
```

   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. The notification response is marked as
   Non-confirmable message. 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.

```
                DOTS Client                DOTS Server
                    |                           |
                    |  GET /<policy-id number>  |
                    |  Token: 0x4a              |    Registration
                    |  Observe: 0               |
                    +-------------------------->|
                    |                           |
                    |  2.05 Content             |
                    |  Token: 0x4a              |    Notification of
                    |  Observe: 12              |    the current state
                    |  status: "mitigation      |
                    |          in progress"     |
                    |<--------------------------+
                    |  2.05 Content             |
                    |  Token: 0x4a              |    Notification upon
                    |  Observe: 44              |     a state change
                    |  status: "mitigation      |
                    |           complete"       |
                    |<--------------------------+
                    |  2.05 Content             |
                    |  Token: 0x4a              |    Notification upon
                    |  Observe: 60              |    a state change
                    |  status: "attack stopped" |
                    |<--------------------------+
                    |                           |
```
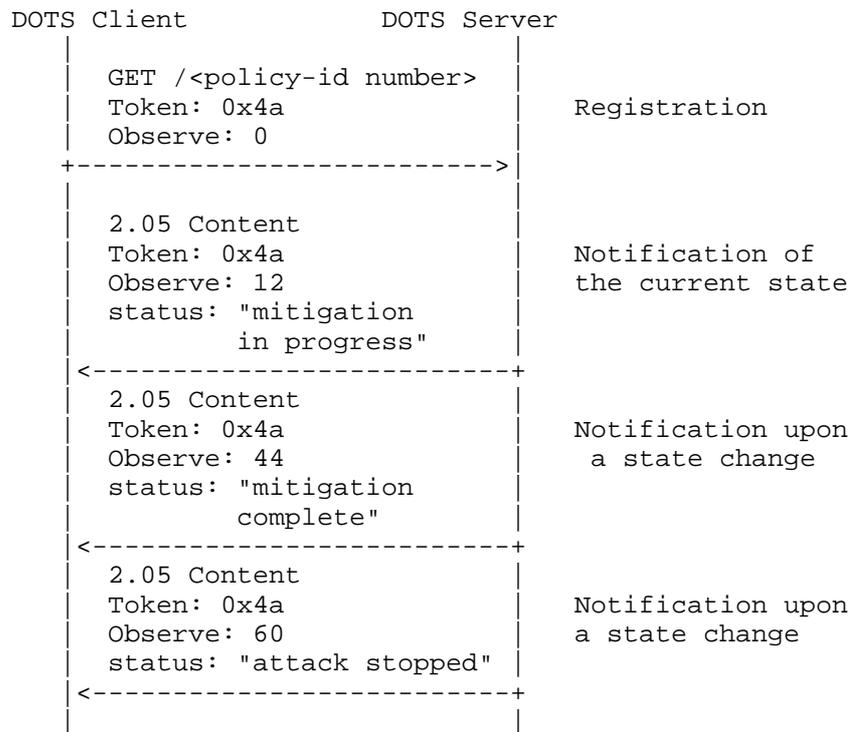
Figure 10: Notifications of attack mitigation status

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). The PUT request and response are marked as Non-
   Confirmable messages.

```
      Header: PUT (Code=0.03)
      Uri-Host: "host"
      Uri-Path: ".well-known"
      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"
              ],
              "E.164": [
                "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 interval: DOTS agents regularly send heartbeats to each
       other after mutual authentication in order to keep the DOTS
       signal channel open.
   b. Acceptable signal loss ratio: Maximum retransmissions,
       retransmission timeout value and other message transmission
       parameters for the DOTS signal channel.

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. The GET request and response are marked as Confirmable
   messages.

```
   Header: GET (Code=0.01)
   Uri-Host: "host"
   Uri-Path: ".well-known"
   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-interval": {"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 interval, 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 guidence 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. The POST request and response are marked as Confirmable
   messages.

```
   Header: POST (Code=0.02)
   Uri-Host: "host"
   Uri-Path: ".well-known"
   Uri-Path: "version"
   Uri-Path: "dots-signal"
   Uri-Path: "config"
   Content-Format: "application/cbor"
   {
    "signal-config": {
      "policy-id": integer,
      "heartbeat-interval": 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-interval: Heartbeat interval 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 intial
      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-interval
   or max-retransmit or ack-timeout or ack-random-factor MUST be
   present. The POST request with higher numeric policy-id value over-
   rides 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: ".well-known"
     Uri-Path: "v1"
     Uri-Path: "dots-signal"
     Uri-Path: "config"
     Content-Format: "application/cbor"
     {
       "signal-config": {
        "policy-id": 1234534333242,
        "heartbeat-interval": 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-interval, 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
   minumum 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-interval": {"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). The DELETE request and
   response are marked as Confirmable messages.

```
     Header: DELETE (Code=0.04)
     Uri-Host: "host"
     Uri-Path: ".well-known"
     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. The
   GET request and response are marked as Confirmable messages.

```
   Header: GET (Code=0.01)
   Uri-Host: "host"
   Uri-Path: ".well-known"
   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 can mark
   the notification response conveying the alternate server address as a
   a Confirmable message to request an acknowledgement from the DOTS
   client.

   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                      |
| E.164             | 11                   | 4                      |
| alias             | 12                   | 4                      |
| lifetime          | 13                   | 0                      |
| attack-status     | 14                   | 0                      |
| signal-config     | 15                   | 5                      |
| heartbeat-interval| 16                   | 0                      |
| max-retransmit    | 17                   | 0                      |
| ack-timeout       | 18                   | 0                      |
| ack-random-factor | 19                   | 7                      |
| MinValue          | 20                   | 0                      |
| MaxValue          | 21                   | 0                      |
| status            | 22                   | 0                      |
| bytes_dropped     | 23                   | 0                      |
| bps_dropped       | 24                   | 0                      |
| pkts_dropped      | 25                   | 0                      |
| pps_dropped       | 26                   | 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.

```
       DOTS Client                                    DOTS Server

      ClientHello
      (Finished)
      (0-RTT DOTS signal message)
      (end_of_early_data)        -------->
                                                         ServerHello
                                                {EncryptedExtensions}
                                                {ServerConfiguration}
                                                        {Certificate}
                                                  {CertificateVerify}
                                                           {Finished}
                                   <--------   [DOTS signal message]
      {Finished}                   -------->

      [DOTS signal message]        <------->   [DOTS signal message]
```

                   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.

```
   +----------------------------------------------------+
   |           example.com domain        +---------+    |
   |                                     |  AAA    |    |
   |     +---------------+               | Server  |    |
   |     |  Application  |               +------+--+    |
   |     |  server       +                      ^       |
   |     |  (DOTS client)|<---------------+      |       |          example.net
   |     +--------------+                 +      |      |          domain
   |                                                   |
domain                                   V   V      |
   |                             +------------+    |          +------------
--+
   |   +-------------+           |            |    |          |
  |
   |   |   Guest      +<-----x----->+           +<--------------->+    DOTS
  |
   |   | (DOTS client)|            |  DOTS     |    |          |   Server
  |
   |   +-------------+            |  Gateway  |    |          |
  |
   |                              +----+-------+    |          +------------
--+
   |                                   ^           |
   |                                   |           |
   |    +---------------+              |           |
   |    | DDOS detector |              |           |
   |    | (DOTS client) +<-------------+           |
   |    +---------------+                          |
   |                                               |
   +-----------------------------------------------+
```
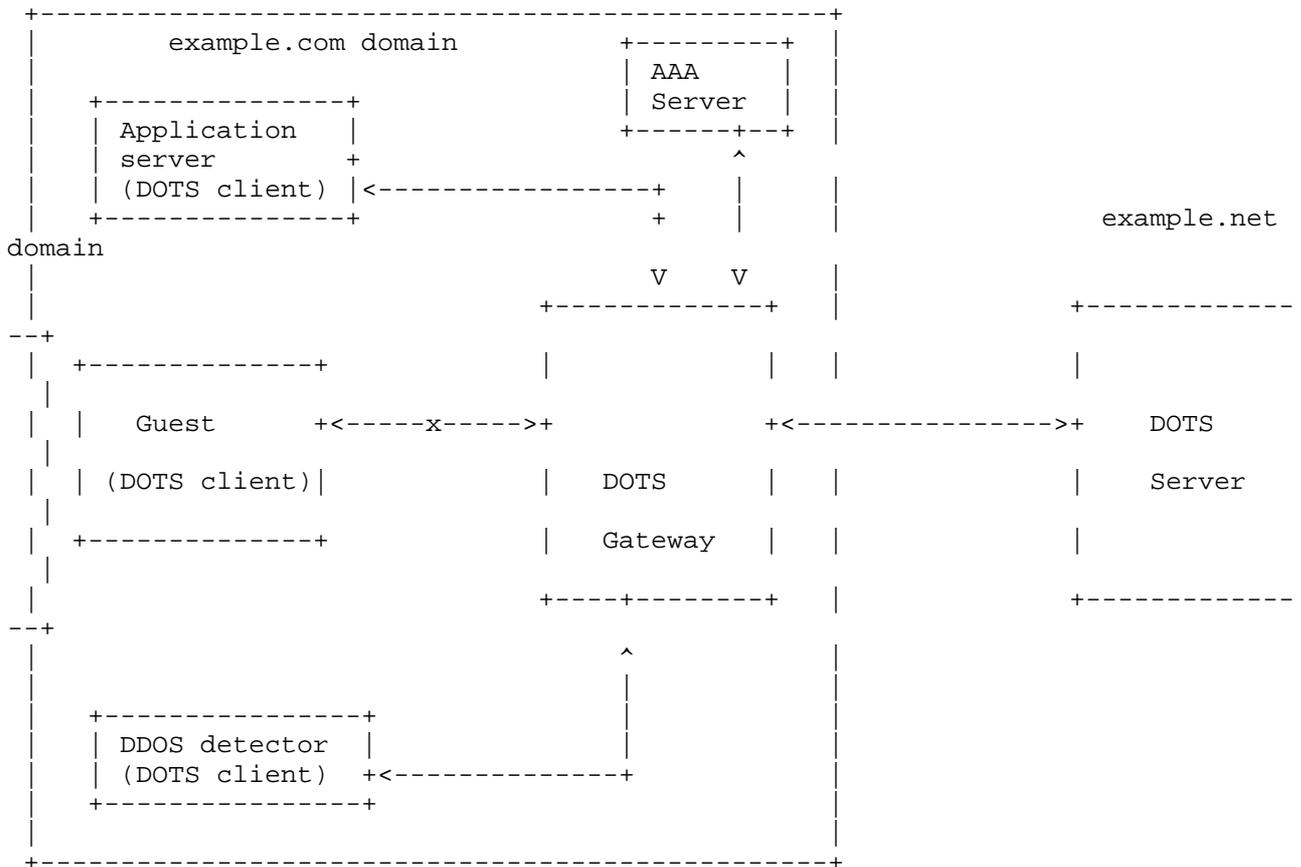
       Figure 23: Example of Authentication and Authorization of DOTS Agents

       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.

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: "E.164"
   o CBOR Key Value: 11
   o CBOR Major Type: 4

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

      o Parameter Name: alias
      o CBOR Key Value: 12
      o CBOR Major Type: 4
      o Change Controller: IESG
      o Specification Document(s): this document

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

      o Parameter Name: attack-status
      o CBOR Key Value: 14
      o CBOR Major Type: 0
      o Change Controller: IESG
      o Specification Document(s): this document

      o Parameter Name: signal-config
      o CBOR Key Value: 15
      o CBOR Major Type: 5
      o Change Controller: IESG
      o Specification Document(s): this document

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

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

      o Parameter Name: ack-timeout
      o CBOR Key Value: 18
      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: 19
      o CBOR Major Type: 7

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

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

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

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

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

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

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

   o Parameter Name: pps_dropped
   o CBOR Key Value: 26
   o CBOR Major Type: 0
   o Change Controller: IESG
   o Specification Document(s): this document

11. 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.

   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.

12. 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

13. 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.

14. References

14.1. 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-06 (work in progress),
             February 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>.

14.2. 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-03 (work in
              progress), October 2016.

   [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-03 (work
              in progress), November 2016.

   [I-D.ietf-tls-tls13]
              Rescorla, E., "The Transport Layer Security (TLS) Protocol
              Version 1.3", draft-ietf-tls-tls13-18 (work in progress),
              October 2016.

   [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-04 (work in progress), February
              2017.

   [I-D.rescorla-tls-dtls13]
              Rescorla, E. and H. Tschofenig, "The Datagram Transport
              Layer Security (DTLS) Protocol Version 1.3", draft-
              rescorla-tls-dtls13-00 (work in progress), October 2016.

   [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>.

   [RFC6520] Seggelmann, R., Tuexen, M., and M. Williams, "Transport
              Layer Security (TLS) and Datagram Transport Layer Security
              (DTLS) Heartbeat Extension", RFC 6520,
              DOI 10.17487/RFC6520, February 2012,
              <http://www.rfc-editor.org/info/rfc6520>.

   [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>.

   [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: tireddy@cisco.com

   Mohamed Boucadair
   Orange
   Rennes 35000
   France

   Email: mohamed.boucadair@orange.com

   Prashanth Patil
   Cisco Systems, Inc.

   Email: praspati@cisco.com