

TMRID
Internet-Draft
Intended status: Informational
Expires: 1 August 2020

S. Card
A. Wiethuechter
AX Enterprize
R. Moskowitz
HTT Consulting
29 January 2020

UAS Remote ID
draft-card-tmrid-uas-01

Abstract

This document is an Applicability Statement for various IETF Technical Specifications, complementing emerging external standards and regulations to meet needs for Unmanned Aircraft System (UAS) remote identification (RID). The objectives are: to facilitate use of existing Internet services to support UAS RID and to enable enhanced RID related services; and to enable verification that UAS RID information is trustworthy (to some extent, even in the absence of Internet connectivity at the receiving node).

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 <https://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 1 August 2020.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights

and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terms and Definitions	3
2.1. Requirements Terminology	4
2.2. Definitions	4
3. UAS RID Problem Space	6
3.1. Network RID	6
3.2. Broadcast RID	7
3.3. TM-RID Focus Problem Space	8
4. Alternatives for IETF work on Trustworthy IDs	8
4.1. Requirements of Trustworthy IDs	8
4.2. Currently selected IDs by ASTM	8
4.3. Options for Trustworthy IDs	9
5. IANA Considerations	9
6. Security Considerations	9
7. Acknowledgments	10
8. References	10
8.1. Normative References	10
8.2. Informative References	10
Authors' Addresses	11

1. Introduction

Emerging Civil Aviation Authority (CAA) regulations worldwide, exemplified by current United States (US) Federal Aviation Administration (FAA) rulemaking, will soon mandate, and many safety and other considerations dictate (even absent regulations), that Unmanned Aircraft Systems (UAS) be remotely identifiable. CAAs are expected and FAA has stated its intent to require compliance with industry consensus standards.

ASTM International, Technical Committee F38 (UAS), Subcommittee F38.02 (Aircraft Operations), Work Item WK65041 (UAS Remote ID and Tracking), is a Proposed New Standard [WK65041]. It defines 2 means of UAS remote identification (RID): Network RID via the Internet; and Broadcast RID via a one-way data link direct from the Unmanned Aircraft (UA) to the observer's device. Network RID depends upon Internet connectivity between the observer and either the UA itself or any of various proxies. Broadcast RID should need Internet (or other Wide Area Network) connectivity only for UAS registry information lookup using the directly locally received UAS ID as a key.

The need for near-universal deployment of UAS RID is pressing. This implies the need to support use by observers of already ubiquitous mobile devices (smartphones and tablets). UA onboard RID devices are severely constrained in Size, Weight and Power (SWaP). Cost is a significant impediment to the necessary near-universal adoption of UAS send and observer receive RID capabilities. To accomodate the most severely constrained cases, all these conspire to motivate system design decisions, especially for the Broadcast RID data link, which complicate the protocol design problem: one-way links; extremely short packets; and Internet-disconnected operation of UA onboard devices. Internet-disconnected operation of observer devices has been deemed by ASTM F38.02 too infrequent to address, but for some users is important and presents further challenges.

Heavyweight security protocols are infeasible, yet trustworthiness of UAS RID information is essential. Even the most basic datum, the UAS ID string (typically number) itself, under [WK65041], can be merely an unsubstantiated claim.

Further, an ID is not an end in itself; it exists to enable lookups and provision of services complementing mere identification, e.g. dynamic establishment of secure communications between the observer and the UAS pilot. [WK65041] neither fully specifies nor appears to facilitate these functions, especially in the case where the observer lacks real time Internet access.

Finally, [WK65041] proposes the use of plaintext and mostly static UAS ID strings. Even if lookup from these to operator Personally Identifiable Information (PII) is successfully limited to strongly authenticated personnel, properly authorized per policy: static IDs enable trivial correlation of patterns of use, unacceptable in many applications, e.g. package delivery routes of competitors.

IETF can help by providing expertise as well as mature and evolving standards. Host Identity Protocol (HIPv2) [RFC7401] and its Domain Name System (DNS) extensions [RFC8005] can complement emerging external standards for UAS RID, to facilitate utilization of existing and provision of enhanced network services, and to enable verification that UAS RID information is trustworthy (to some extent, even in the absence of Internet connectivity at the receiving node).

2. Terms and Definitions

2.1. Requirements Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2.2. Definitions

CAA

Civil Aviation Authority. An example is the Federal Aviation Administration (FAA) in the United States of America.

C2

Command and Control. A set of organizational and technical attributes and processes that employs human, physical, and information resources to solve problems and accomplish missions. Mainly used in military contexts.

GCS

Ground Control Station. The part of the UAS that the remote pilot uses to exercise C2 over the UA, whether by remotely exercising UA flight controls to fly the UA, by setting GPS waypoints, or otherwise directing its flight.

HI

Host Identity. The public key portion of an asymmetric keypair from HIP. In this document it is assumed that the HI is based on a EdDSA25519 keypair. This is supported by new crypto defined in [I-D.moskowitz-hip-new-crypto].

HIT

Host Identity Tag. A 128 bit handle on the HI. Defined in HIPv2 [RFC7401].

HHIT

Hierarchical Host Identity Tag. A HIT with extra information not found in a standard HIT. Defined in [I-D.moskowitz-hip-hierarchical-hit].

UA

Unmanned Aircraft. Typically a military or commercial "drone" but can include any and all aircraft that are unmanned.

UAS

Unmanned Aircraft System. Composed of UA, all required on-board subsystems, payload, control station, other required off-board

subsystems, any required launch and recovery equipment, all required crew members, and C2 links between UA and control station.

UTM

UAS Traffic Management. A "traffic management" ecosystem for "uncontrolled" UAS operations separate from, but complementary to, the FAA's Air Traffic Management (ATM) system for "controlled" operations of manned aircraft.

USS

UAS Service Supplier. Provide UTM services to support the UAS community, to connect Operators and other entities to enable information flow across the USS network, and to promote shared situational awareness among UTM participants. (From FAA UTM ConOps V1, May 2018).

RID

Remote ID. System for identifying UA during flight by other parties.

Observer

Referred to in other UAS documents as a "user", but there are also other classes of RID users, so we prefer "observer" to denote an individual who has observed an UA and wishes to know something about it, starting with its ID.

UAS ID

Unique UAS identifier. Per [WK65041], maximum length of 20 bytes.

UAS ID Type

Identifier type index. Per [WK65041], 4 bits, values 0-3 already specified.

RID SP

UAS RID Service Provider. System component that compiles information from various sources (and methods) in its given service area.

RID DP

UAS RID Display Provider. System component that requests data from one or more RID SP and aggregates them to display to a user application on a device.

UAS RID Verification Service

System component designed to handle the authentication requirements of RID by offloading verification to a web hosted service.

3. UAS RID Problem Space

UA may be fixed wing Short Take-Off and Landing (STOL), rotary wing (e.g. helicopter) Vertical Take-Off and Landing (VTOL), or hybrid. They may be single engine or multi engine. The most common today are multicopters: rotary wing, multi engine. The explosion in UAS was enabled by hobbyist development, for multicopters, of advanced flight stability algorithms, enabling even inexperienced pilots to take off, fly to a location of interest, hover, and return to the take-off location or land at a distance. UAS can be remotely piloted by a human (e.g. with a joystick) or programmed to proceed from Global Positioning System (GPS) waypoint to waypoint in a weak form of autonomy; stronger autonomy is coming. UA are "low observable": they typically have a small radar cross section; they make noise quite noticeable at short range but difficult to detect at distances they can quickly close (500 meters in under 17 seconds at 60 knots); they typically fly at low altitudes (for the small UAS to which RID applies, under 400 feet Above Ground Level in the US); they are highly maneuverable so can fly under trees and between buildings.

UA can carry payloads including sensors, cyber and kinetic weapons or can be used themselves as weapons by flying them into targets. They can be flown by clueless, careless or criminal operators. Thus the most basic function of UAS RID is "Identification Friend or Foe" to mitigate the significant threat they present. Numerous other applications can be enabled or facilitated by RID: consider the importance of identifiers in many Internet protocols and services.

Network RID from the UA itself (rather than from a proxy) and Broadcast RID require one or more wireless data links from the UA, but such communications are challenging due to \$SWaP constraints and low altitude flight amidst structures and foliage over terrain.

3.1. Network RID

Network RID has several variants. The UA may have persistent onboard Internet connectivity, in which case it can consistently source RID information directly over the Internet. The UA may have intermittent onboard Internet connectivity, in which case a proxy must source RID information whenever the UA itself is offline. The UA may not have Internet connectivity of its own, but have instead some other form of communications to a (typically ground) node that can relay RID information to the Internet; this would typically be the GCS (which to perform its function must know where the UA is) or USS (which in the UTM system is required to be kept informed by the UAS operator). The UA may have no means of sourcing RID information, in which case the GCS, USS or other proxy may source it. In the extreme case, this would be the pilot using a web browser to designate, to a USS or

other UTM entity, a time-bounded airspace volume in which an operation will be conducted; this may impede disambiguation of ID if multiple UAS operate in the same or overlapping spatio-temporal volumes.

In most cases in the near term, if the RID information is fed to the Internet directly by the UA or remote pilot, the first hop data links will be cellular Long Term Evolution (LTE) or WiFi, but provided the data link can support at least IP and ideally TCP, its type is generally immaterial to the higher layer protocols. The ultimate source of Network RID information feeds a RID Service Provider (SP), which essentially proxies for that and other sources; the ultimate consumer of Network RID information obtains it from a RID Display Provider (DP). Each DP aggregates information from all SPs that have UA currently operating in the airspace for which that DP is cognizant.

Network RID is the more flexible and less constrained of the UAS RID means specified in [WK65041]. Any IETF work needed to support or leverage it is left for later efforts; it is not further addressed herein or in other initial tm-rid documents.

3.2. Broadcast RID

[WK65041] specifies 3 Broadcast RID data links: Bluetooth 4.X; Bluetooth 5.X Long Range; and Wifi with Neighbor Awareness Networking (NAN). For compliance with this standard, an UA must broadcast (using advertisement mechanisms where no other option supports broadcast) on at least one of these; if broadcasting on Bluetooth 5.x, it is also required concurrently to do so on 4.x (referred to in [WK65041] as Bluetooth Legacy).

The selection of the Broadcast medium was driven by research into what is commonly available on 'ground' units (smartphones and tablets) and what was found as prevalent or 'affordable' in UA. Further, there must be an API for the UAS receiving application to have access to these messages. At this time, only Bluetooth 4.X support is readily available, thus the current focus is on working within the 26 byte limit of the Bluetooth 4.X "Broadcast Frame" that goes out on the beacon channels.

Finally, the 26 byte limit of the Bluetooth 4.1 "Broadcast Frame" strictly enforces the RID maximum length of 20 bytes.

3.3. TM-RID Focus Problem Space

TM-RID will focus on adding immediate usability, thus trust to, Broadcast RID. The one-way nature of Broadcast RID precludes any stateful security protocol. Under [WK65041], any UA can announce a RID and an observer would be seriously challenged to validate it or any other information about the UA looked up from it. Thus providing trust in the RID and related trust for all Broadcast messages is critical for the safe and secure operation of UAs.

Three levels of functionality will be considered:

1. verify that HHIT is duly registered with a known registry AND that any messages signed with its key came from it;
2. look up not only static UAS registry and dynamic UTM information but also Internet direct contact information for services relating to the UA, its current mission, etc., including communications with the remote pilot (or proxy) and USS;
3. dynamically establish strongly mutually authenticated, E2E strongly encrypted communications with the UAS RID sender and entities looked up via (2) above.

4. Alternatives for IETF work on Trustworthy IDs

4.1. Requirements of Trustworthy IDs

Just a couple of requirements:

1. The ID MUST be 20 bytes or smaller.
2. It MUST be non-spoofable within the context of Remote ID broadcast messages (some collection of messages provides proof of UA ownership of ID).
3. In context (that is in a Remote ID Broadcast message), just the ID provides enough information on how at least the observer's USS (UAS Service Provider / Display Provider) can provide both public and private information on the UAS.

4.2. Currently selected IDs by ASTM

Now a little 'context' setting. ASTM has already defined a set of textual Remote IDs:

- 1 Serial Number [CTA2063A]

2 CAA Assigned ID

3 UTM Assigned ID [RFC4122]

The work here MUST surpass these in terms of Trustworthiness.

4.3. Options for Trustworthy IDs

The options found are:

1. X.509 certs where something like the cert sequenceNumber is the Remote ID.
2. Naming Things with Hashes, Section 8.2 of [RFC6920]
3. SSH keyID
4. HIT (Host Identity Tag) [RFC7401]

Option 1 is no better than what ASTM/FAA is considering for any of the current proposed types. Somehow, there will be a PKI and from that knowledge of the UAS is gained. This REQUIRES Internet Access (think disaster or other non-Internet situations) and a GLOBAL PKI (the UA flew from Canada to the US or UK to France post Brexit).

Option 2 meets requirements 1 and 2, but needs to be augmented so that the Hash provides context for 3. Is it supported for IPsec and/or QUIC for UAS/observer secure communications (NetworkID).

5. IANA Considerations

It is likely that an IPv6 prefix will be needed for the HHIT (or other identifier) space; this will be specified in other drafts.

6. Security Considerations

UAS RID is all about safety and security, so content pertaining to such is not limited to this section. UAS RID information must be divided into 2 classes: that which, to achieve the purpose, must be published openly in plaintext, for the benefit of any observer; and that which must be protected (e.g. PII of pilots) but made available to properly authorized parties (e.g. public safety personnel who urgently need to contact pilots in emergencies). Details of the protection mechanisms will be provided in other drafts. Classifying the information will be addressed primarily in external standards but also herein as needed.

7. Acknowledgments

The work of the FAA's UAS Identification and Tracking (UAS ID) Aviation Rulemaking Committee (ARC) is the foundation of later ASTM and proposed IETF efforts. The work of ASTM F38.02 in balancing the interests of diverse stakeholders is essential to the necessary rapid and widespread deployment of UAS RID.

8. References

8.1. 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>>.
- [RFC7401] Moskowitz, R., Ed., Heer, T., Jokela, P., and T. Henderson, "Host Identity Protocol Version 2 (HIPv2)", RFC 7401, DOI 10.17487/RFC7401, April 2015, <<https://www.rfc-editor.org/info/rfc7401>>.
- [RFC8005] Laganier, J., "Host Identity Protocol (HIP) Domain Name System (DNS) Extension", RFC 8005, DOI 10.17487/RFC8005, October 2016, <<https://www.rfc-editor.org/info/rfc8005>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

8.2. Informative References

- [CTA2063A] ANSI, "Small Unmanned Aerial Systems Serial Numbers", September 2019.
- [I-D.moskowitz-hip-hierarchical-hit]
Moskowitz, R., Card, S., and A. Wiethuechter, "Hierarchical HITs for HIPv2", Work in Progress, Internet-Draft, draft-moskowitz-hip-hierarchical-hit-03, 16 December 2019, <<https://tools.ietf.org/html/draft-moskowitz-hip-hierarchical-hit-03>>.
- [I-D.moskowitz-hip-new-crypto]
Moskowitz, R., Card, S., and A. Wiethuechter, "New Cryptographic Algorithms for HIP", Work in Progress, Internet-Draft, draft-moskowitz-hip-new-crypto-04, 23 January 2020, <<https://tools.ietf.org/html/draft-moskowitz-hip-new-crypto-04>>.

- [RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique Identifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <<https://www.rfc-editor.org/info/rfc4122>>.
- [RFC6920] Farrell, S., Kutscher, D., Dannewitz, C., Ohlman, B., Keranen, A., and P. Hallam-Baker, "Naming Things with Hashes", RFC 6920, DOI 10.17487/RFC6920, April 2013, <<https://www.rfc-editor.org/info/rfc6920>>.
- [WK65041] ASTM, "Standard Specification for Remote ID and Tracking", September 2019.

Authors' Addresses

Stuart W. Card
AX Enterprize
4947 Commercial Drive
Yorkville, NY 13495
United States of America

Email: stu.card@axenterprize.com

Adam Wiethuechter
AX Enterprize
4947 Commercial Drive
Yorkville, NY 13495
United States of America

Email: adam.wiethuechter@axenterprize.com

Robert Moskowitz
HTT Consulting
Oak Park, MI 48237
United States of America

Email: rgm@labs.htt-consult.com

TSVWG
Internet-Draft
Intended status: Experimental
Expires: May 7, 2020

M. Cociglio
Telecom Italia
G. Fioccola
Huawei Technologies
F. Bulgarella
R. Sisto
Politecnico di Torino
November 4, 2019

New Spin bit enabled measurements with one or two more bits
draft-cfb-tsvwg-spinbit-new-measurements-00

Abstract

This document introduces additional measurements by using the same spin bit signal as defined in [I-D.trammell-tsvwg-spin] and [I-D.trammell-ippm-spin]. The spin bit signal alone is not enough to evaluate correctly in every network condition the RTT of a flow. In order to solve this problem, it is theorized the possibility of introducing an additional validation signal called delay bit, similar to what is done by the Valid Edge Counter (VEC), but using just one bit instead of two. An alternative with two bits is also introduced with a so called loss bit. More in general a loss signal is defined to measure packet loss and two alternatives are presented with one bit and two bits.

Requirements Language

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].

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 <https://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 7, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Spin bit and Delay bit mechanism	3
2.1. Delay Sample generation	5
2.1.1. The recovery process	6
2.2. Delay Sample reflection	6
3. Using the Spin bit and Delay bit for Hybrid RTT Measurement	7
3.1. End-to-end RTT measurement	7
3.2. Half-RTT measurement	7
3.3. Intra-domain RTT measurement	8
4. Observer's algorithm and Waiting Interval	8
5. Adding a Loss signal for Packet loss measurement	9
5.1. Round Trip Packet Loss measurement	10
6. RTT dependent Packet Loss using one bit loss signal	11
6.1. Observer's logic for one bit loss signal	12
7. RTT independent Packet Loss using two bits loss signal	12
7.1. Observer's logic for two bits loss signal	14
8. Protocols	14
8.1. QUIC	14
8.2. TCP	15
9. Security Considerations	15
10. Acknowledgements	15
11. IANA Considerations	15
12. References	15
12.1. Normative References	15
12.2. Informative References	16
Authors' Addresses	16

1. Introduction

Both [I-D.trammell-tsvwg-spin] and [I-D.trammell-ippm-spin] define an explicit per-flow transport-layer signal for hybrid measurement of end-to-end RTT. This signal consists of three bits: a spin bit, which oscillates once per end-to-end RTT, and a two-bit Valid Edge Counter (VEC), which compensates for loss and reordering of the spin bit to increase fidelity of the signal in less than ideal network conditions.

In this document it is introduced the delay bit, that is a single bit signal that can be used together with the spin bit by passive observers to measure the RTT of a network flow, avoiding the spin bit ambiguities that arise as soon as network conditions deteriorate. Unlike the spin bit, which is actually set in every packet transmitted on the network, the delay bit is set only once per round trip.

This document defines a hybrid measurement RFC 7799 [RFC7799] path signal to be embedded into a transport layer protocol, explicitly intended for exposing end-to-end RTT to measurement devices on path.

The document introduces a mechanism applicable to any transport-layer protocol, then explains how to bind the signal to a variety of IETF transport protocols, and in particular to QUIC and TCP.

The application of the Spin bit to QUIC is described in [I-D.ietf-quic-spin-exp] which adds the spin bit only (without the VEC) to QUIC for experimentation purposes.

Note that both the spin bit and the delay bit are inspired by RFC 8321 [RFC8321]. This is also mentioned in [I-D.trammell-quic-spin].

2. Spin bit and Delay bit mechanism

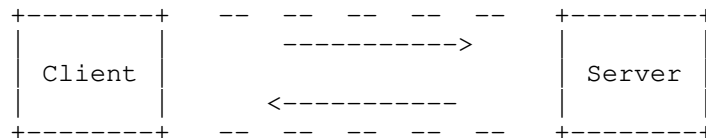
The main idea is to have a single packet, with a second marked bit (the delay bit), that bounces between client and server during the entire connection life. This single packet is called Delay Sample.

A simple observer placed in an intermediate point, tracking the delay sample and the relative timestamp in every spin bit period, can measure the end-to-end round trip delay of the connection. In the same way as seen with the spin bit and the VEC, it is possible to carry out other types of measurements. The next paragraphs give an overview of the observer capabilities.

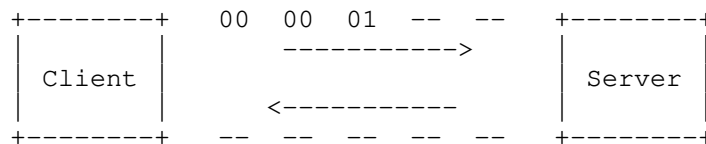
In order to describe the delay sample working mechanism in detail, we have to distinguish two different phases which take part in the delay

bit lifetime: initialization and reflection. The initialization is the generation of the delay sample, while the reflection realizes the bounce behavior of this single packet between the two endpoints.

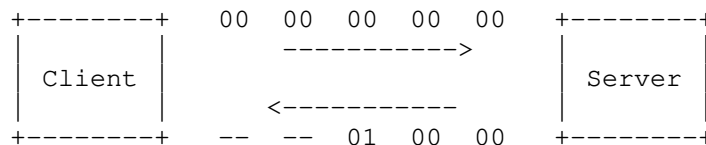
The next figure describes the Delay bit mechanism: the first bit is the spin bit and the second one is the delay bit.



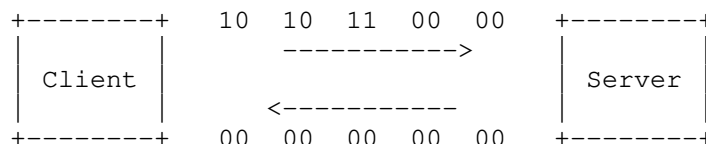
(a) No traffic at beginning.



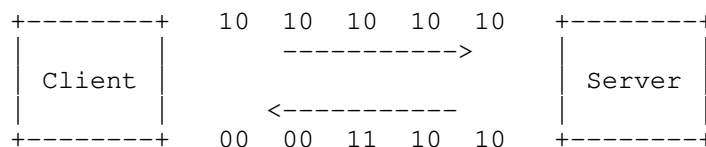
(b) The Client starts sending data and sets the first packet as Delay Sample.



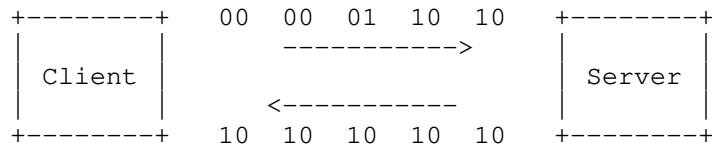
(c) The Server starts sending data and reflects the Delay Sample.



(d) The Client inverts the spin bit and reflects the Delay Sample.



(e) The Server reflects the Delay Sample.



(f) The client reverts the spin bit and reflects the Delay Sample.

Figure 1: Spin bit and Delay bit

2.1. Delay Sample generation

During this first phase, endpoints play different roles. First of all a single delay sample must be bouncing per round trip period (and so per spin bit period). According to that statement and in order to simplify the general algorithm, the delay sample generation is in charge of just one of the two endpoints:

- o the Client, when connection starts and spin bit is set to 0, initializes the delay bit of the first packet to 1, so it becomes the delay sample for that marking period. Only this packet is marked with the delay bit set to 1 for this round trip period; the other ones will carry only the spin bit;
- o the server never initializes the delay bit to 1; its only task is to reflect the incoming delay bit into the next outgoing packet only if certain conditions occur.

Theoretically, in absence of network impairments, the delay sample should bounce between client and server continuously, for the entire duration of the connection. Actually, that is highly unlikely mainly for two different reasons:

- 1) the packet carrying the delay bit might be lost during its journey on the network which is unreliable by definition;
- 2) one of the two endpoints could stop or delay sending data because the application is limiting the amount of traffic transmitted;

To deal with these problems, the algorithm provides a procedure to regenerate the delay sample and to inform a possible observer that a problem has occurred, and then the measurement has to be restarted.

2.1.1. The recovery process

In order to relieve the server from tasks that go beyond the mere reflection of the sample, even in this case the recovery process belongs to the client. A fundamental assumption is that a delay sample is strictly related to its spin bit period. Considering this rule, the client verifies that every spin bit period ends with its delay sample. If that does not happen and a marking period terminates without a delay sample, the client waits a further empty period; then, in the following period, it reinitializes the mechanism by setting the delay bit of the first outgoing packet to 1, making it the new delay sample. The empty period is needed to inform the intermediate points that there was an issue and a new delay measurement session is starting.

2.2. Delay Sample reflection

The reflection is the process that enables the bouncing of the delay sample between client and server. The behavior of the two endpoints is slightly different. With the exception of the client that, as previously exposed, generates a new delay sample, by default the delay bit is set to 0.

Server side reflection: when a packet with the delay bit set to 1 arrives, the server marks the first packet in the opposite direction as the delay sample, if it has the same spin bit value. While if it has the opposite spin bit value this sample is considered lost.

Client side reflection: when a packet with delay bit set to 1 arrives, the client marks the first packet in the opposite direction as the delay sample, if it has the opposite spin bit value. While if it has the same spin bit value this sample is considered lost.

In both cases, if the outgoing marked packet is transmitted with a delay greater than a predetermined threshold after the reception of the incoming delay sample (lms by default), reflection is aborted and this sample is considered lost.

It is noteworthy that differently from what happens with the VEC for which the reflection always concerns the edge of the period, in this case reflection takes place for the packet that is carrying the delay bit regardless of its position within the period. For this reason it is necessary to introduce that condition of validation in order to identify and discard those samples that, due to reordering, might move to a contiguous period. Furthermore, by introducing a threshold for the retransmission delay of the sample, it is possible to eliminate all those measurements which, due to lack of traffic on the endpoints, would be overestimated and not true. Thus, the maximum

estimation error, without considering any other delays due to flow control, would amount to twice the threshold (e.g. 2ms) per measurement, in the worst case.

3. Using the Spin bit and Delay bit for Hybrid RTT Measurement

Unlike what happens with the spin bit for which it is necessary to validate or at least heuristically evaluate the goodness of an edge, the delay sample can be used by an intermediate observer as a simple demarcator between a period and the following one eliminating the ambiguities on the calculation of the RTT found with the analysis of the spin-bit only. The measurement types, that can be done from the observation of the delay sample, are exactly the same achievable with the spin bit only (with or without the VEC).

3.1. End-to-end RTT measurement

The delay sample generation process ensures that only one packet marked with the delay bit set to 1 runs back and forth on the wire between two endpoints per round trip time. Therefore, in order to determine the end-to-end RTT measurement of a QUIC flow, an on-path passive observer can simply compute the time difference between two delay samples observed in a single direction. Note that a measurement, to be valid, must take into account the difference in time between the timestamps of two consecutive delay samples belonging to adjacent spin-bit periods. For this reason, an observer, in addition to intercepting and analyzing the packets containing the delay bit set to 1, must maintain awareness of each spin period in such a way as to be able to assign each delay sample to its period and, at the same time, identifying those periods that do not contain it.

3.2. Half-RTT measurement

An on-path passive observer that is sniffing traffic in both directions -- from client to server and from server to client -- can also use the delay sample to measure "upstream" and "downstream" RTT components. Also known as the half-RTT measurement, it represents the components of the end-to-end RTT concerning the paths between the client and the observer (upstream), and the observer and the server (downstream). It does this by measuring the delay between a delay sample observed in the downstream direction and the one observed in the upstream direction, and vice versa. Also in this case, it should verify that the two delay samples belong to two adjacent periods, for the upstream component, or to the same period for the downstream component.

3.3. Intra-domain RTT measurement

Taking advantage of the half-RTT measurements it is also possible to calculate the intra-domain RTT which is the portion of the entire RTT used by a QUIC flow to traverse the network of a provider (or part of it). To achieve this result two observers, able to watch traffic in both directions, must be employed simultaneously at ingress and egress of the network to be measured. At this point, to determine the delay between the two observers, it is enough to subtract the two computed upstream (or downstream) RTT components.

The spin bit is an alternate marking generated signal and the only difference than RFC 8321 [RFC8321] is the size of the alternation that will change with the flight size each RTT. So it can be useful to segment the RTT and deduce the contribution to the RTT of the portion of the network between two on-path observers and it can be easily performed by calculating the delay between two or more measurement points on a single direction by applying RFC 8321 [RFC8321].

4. Observer's algorithm and Waiting Interval

Given below is a formal summary of the functioning of the observer every time a delay sample is detected. A packet containing the delay bit set to 1:

- o if it has the same spin bit value of the current period and no delay sample was detected in the previous period, then it can be used as a left edge (i.e. to start measuring an RTT sample), but not as a right edge (i.e. to complete an RTT measurement since the last edge). If the observation point is symmetric (i.e. it can see both upstream and downstream packets in the flow) and in the current period a delay sample was detected in the opposite direction (i.e. in the upstream direction), the packet can also be used to compute the downstream RTT component.
- o if it has the same spin bit value of the current period and a delay sample was detected in the previous period, then it can be used at the same time as a left or right edge, and to compute RTT component in both directions.

Like stated previously, every time an empty period is detected, the observer must restart the measurement process and consider the next delay sample that will come as the beginning of a new measure, then as a left edge. As a result, being able to assign the delay sample to the corresponding spin period becomes a crucial factor for the proper functioning of the entire algorithm.

Considering that the division into periods is realized by exploiting the spin bit square wave, it is easy to understand that the presence of spurious spin edges -- caused by packet reordering -- would inevitably lead the observer to overestimate the amount of periods actually present in the transmission. This results in a greater number of empty periods detected and the consequent decrease of the actual RTT samples achievable. Therefore, in order to maximize the performance of the whole algorithm, the observer must implement a mechanism to filter out spurious spin edges.

To face this problem the waiting interval has to be introduced. Basically, every time a spin bit edge is detected, the observer sets a time interval during which it rejects every potential spurious edges observed on the wire. While, at the end of the interval it starts again to accept changes in the spin bit value. This guarantees a proper protection against the spurious edges in relation to the size of the interval itself. For instance, an interval of 5ms is able to filter out edges that have been reordered by a maximum of 5ms. Clearly, the mechanism does its job for intervals smaller than the RTT of the observed connection (if RTT is smaller than the waiting interval the observer can't measure the RTT).

5. Adding a Loss signal for Packet loss measurement

It is possible to introduce a mechanism to evaluate also the packet loss together with the delay measurement. This can be achieved by introducing the loss signal, a single or two bits signal whose purpose is to mark a variable number of packets (from live traffic) which are exchanged two times between the endpoints realizing a two round-trip reflection. The overall exchange comprises:

- o The client first selects, generates and consequent transmits to the server a first train of packets, by marking the loss bit to 1;
- o The server, upon reception from the client of each one of the packets included in the first train, reflects to the client a respective second train of packets of the same size as the first train received, by marking the loss bit to 1;
- o The client, upon reception from the server of each one of the packets included in the second train, reflects to the server a respective third train of packets of the same size as the second train received, by marking the loss bit to 1;
- o The server, upon reception from the client of each one of the packets included in the third train, finally reflects to the client a respective fourth train of packets of the same size as the third train received, by marking the loss bit to 1.

Packets belonging to the first round (first and second train) represent the Generation Phase while those belonging to the second round (third and fourth train) represent the Reflection Phase.

A passive on-path observer, placed on whatever direction, can trivially count and compare the number of marked packets seen during the two mentioned phases (i.e. the first and third or the second and the fourth trains of packets, depending on which direction is observed) and estimate the loss rate experienced by the connection. This process is repeated continuously to obtain more measurements as long as the endpoints exchange traffic. These measurements can be called Round Trip (RT) losses

The general algorithm shown above gives an idea of its underlying principles but is not enough to make the whole process working properly.

Firstly, there is the issue that packet rates in the two directions may be different. Therefore, the right number of packets to be marked has to be chosen in order to avoid their congestion on the slowest traffic direction. As a consequence, this number is inevitably equal to the amount of packets transited, indeed, on the slowest direction. This problem can be easily addressed by a method wherein the two endpoints of a communication exchange marked packets interleaved with unmarked packets. From an implementation point of view, this result can be achieved by introducing a single token system that adjusts the number of outgoing marked packets. Basically, the token is enabled every time a packet arrives and disabled when a marked packet is transmitted. Since the creation of the initial train of marked packets is carried out by the client, the management and use of this single token is also assigned to it, which in fact "calculates" the correct number of packets to be marked each time.

Secondly, a mechanism to individually identify each train of packets must be provided to enable the observer to distinguish between trains belonging to different phases (Generation and Reflection). About this point, different approaches are used depending on the number of bits of the loss signal and it will be discussed in the next sections.

5.1. Round Trip Packet Loss measurement

Since the measurements are performed on a portion of the traffic exchanged between client and server, the observer calculates the end-to-end Round Trip Packet Loss that, statistically, will be equal to the loss rate experienced by the connection along the entire network

path. So this measurement can be simply referred as the Round Trip Packet Loss (RTPL).

In addition, this methodology allows the Half-RTPL measurement and the Intra-domain RTPL measurement, in the same way as described in the previous sections for RTT measurement.

6. RTT dependent Packet Loss using one bit loss signal

The single bit loss signal is implemented using just one bit: marked packets have this bit set to 1, whereas unmarked ones have it set to 0. This solution requires a working spin-bit signal used to separate different trains of packets. In particular, a "pause" of at least one empty spin-bit period is introduced between each phase of the algorithm. An on-path observer can determine in this way if a phase (and therefore a train of packets) is ended and a new one is starting.

The client is in charge of almost the entire complexity of the algorithm. Its task can be summarized in 4 different points:

1. The client starts generating marked packets for two consecutive spin-bit periods; it maintains a generation token that is enabled every time a packet arrives and disabled when another one is forwarded. When this token is disabled, the generation process is paused (i.e. outgoing packets are transmitted unmarked) and resumes as soon as its value returns true, and that happens as soon as a packet is received. In addition, at the end of the first spin-bit period spent in generation, the reflection counter is unlocked to start counting incoming marked packets which will be later reflected;
2. When the generation is completed, the client waits to see in input an empty spin-bit period so as to be sure that everyone has seen at least that empty period. This one will be used by the observer as a divider between generated and reflected packets. During this phase, all the outgoing packets are forwarded with the loss bit set to 0. The reflection counter is still incremented every time a marked packet arrives;
3. The client starts reflecting marked packets until the reflection counter is zeroed; the generation token is also used (in the same way) during this phase to avoid congestion on the slowest traffic direction. In addition, at the end of the first spin-period spent in reflection, the reflection counter is locked to avoid incoming reflected packets incrementing it;

4. When the reflection is completed, the client waits to see in input an empty spin-bit period so as to be sure that everyone has seen at least that empty period. This one will be used by the observer as a divider between reflected and newly generated packets. During this phase, all the outgoing packets are forwarded with the loss bit set to 0. The whole process restarts going back to the first point.

As previously anticipated, the server simply reflects each incoming marked packet sent by the client. It maintains a simple counter that is incremented every time a marked packet arrives and decremented when a marked one is sent in the opposite direction.

This one bit loss signal methodology replies and exposes the RTT of the connection on the wire in any case, when the spin bit and the delay bit are used and when these are disabled.

6.1. Observer's logic for one bit loss signal

The on-path observer, placed in any direction, counts marked packets and separates different trains detecting empty spin-bit periods between them (one or more). Then, it simply computes the difference between a Generation train and a Reflection train to produce a statistical measurement of the Round Trip Packet Loss (RTPL) and of the connection end-to-end loss rate.

Here is an example. Packets are represented by two digits (first one is the spin bit, second one is the loss bit):

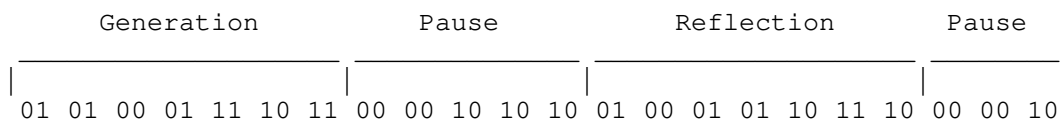


Figure 2: one bit loss signal example

Note that 5 marked packets have been generated of which 4 reflected.

7. RTT independent Packet Loss using two bits loss signal

An RTT independent version of this algorithm requires two bits and does not rely on the spin-bit signal to enable pause detection. That is because packets generated and reflected by the client are marked using two different marking values thus removing the need of introducing a pause between them. Furthermore, instead of generating marked packets for the duration of two spin-bit periods (as seen in

the one bit loss signal), a fixed duration for the generation phase can be used (e.g., 100ms).

In this way, no information related to the RTT of the connection is exposed on wire.

Using a two bits loss signal, four possible values can be used inside each packet (i.e. 0 to 3). During the Generation phase, marked packets have the loss value set to 1 whereas unmarked ones to 0. On the contrary, during the Reflection phase, marked packets have the loss value set to 2 whereas unmarked ones to 3. By doing so, even unmarked packets have their own alternate marking methodology that can be used by intermediate points to compute the one-way loss rate between them (RFC 8321 [RFC8321]).

Even in this case, the client is in charge of almost the entire complexity of the algorithm. Its task can be summarized in 2 different points:

1. The client generates marked packets (i.e. with loss bits set to 1) for 100ms; it also maintains a generation token that is enabled every time a packet arrives and disabled when another one is forwarded. When this token is disabled, the generation process is paused (i.e. outgoing packets are transmitted unmarked with the loss bits set to 0) and resumes as soon as its value returns true.
2. When the generation is completed, the client starts reflecting marked packets (i.e. with loss bits set to 2) until the reflection counter is zeroed and for at least 100ms. The generation token is also used during this phase to avoid congestion on the slowest traffic direction; however, in this case, "unmarked" packets are transmitted with the loss bit set to 3. The whole process restarts going back to the first point.

Independently of the current phase of the algorithm, the reflection counter is increased every time a packet carrying a loss value equal to 1 arrives. Moreover, depending on the connection RTT, the client should vary the duration of the generation phase to different values. For example, for connection below 100ms of RTT the client generates for 100ms; for connection below 300ms of RTT it generates for 300ms and for connection below 1s of RTT it generates for 1000ms. This is necessary to ensure that the client has already received generated marked packets before the beginning of the reflection phase.

As regards the role of the server, it simply reflects each incoming marked packet sent by the client. It maintains two different counters for generated and reflected packets (i.e. loss bits to 1 and

2) in concomitance with a mechanism to reflect in output the same number of marked packets in the same order of arrival (with at most the reordering of packets arrived out of sequence).

7.1. Observer's logic for two bits loss signal

The on-path observer, placed in any direction, counts marked packets belonging to different phases simply looking at the loss value carried by each packet (therefore, it does not look at the spin-bit value anymore). Then, in the same way seen for the previous one bit algorithm, it simply computes the difference between a Generation train and a Reflection train to produce a statistical measurement of the Round Trip Packet Loss (RTPL) and of the connection end-to-end loss rate. Moreover, it can also count unmarked packets and, cooperating with a second observer placed in the same direction, compute the one-way loss rate between two intermediate points using the alternate marking methodology (RFC 8321 [RFC8321]).

Here is an example. Packets are represented by a single digit corresponding to the carried two-bits loss value (0 to 3):

Generation	Reflection	Generation
1 1 0 1 1 1 1 0 1 1 0	2 2 2 2 3 2 3 3 2 3 3	1 1 0 1 0 0 1 1 0 1 0

Figure 3: two bits loss signal example

Note that 8 marked packets have been generated of which 6 reflected; then again 6 marked packets are generated.

8. Protocols

8.1. QUIC

The binding of the delay bit signal to QUIC is partially described in [I-D.ietf-quic-spin-exp], which adds the spin bit only to the QUIC protocol. From an implementation point of view, the delay bit is placed in the partially unencrypted (but authenticated) QUIC header, alongside the spin bit, occupying one of the two bits left reserved for future experiments. As things stand, according to [I-D.ietf-quic-transport], the proposed scheme of the first header's byte would be 01SDRKPP.

Regarding the loss signal, since the use of the spin bit is not mandatory and many connection may not have it spinning, two different configuration are proposed:

If the spin-bit IS enabled (i.e. the RTT is already exposed on wire), use the 1 bit loss signal alongside the delay bit to improve delay measurements accuracy; in this configuration, the proposed scheme of the first header's byte would be 01SDLKPP;

If the spin-bit IS NOT enabled, use the 2 bits loss signal just to measure connection loss rate without exposing any RTT related information on wire; in this configuration, the proposed scheme of the first header's byte would be 01SLLKPP.

This implies that an observer must be able to determine whether the spin bit is active and correctly spinning or not (choosing, accordingly, the right version of packet loss measurement to be used).

8.2. TCP

The signal can be added to TCP by defining bit 4 of bytes 13-14 of the TCP header to carry the spin bit, and eventually bits 5 and 6 to carry additional information, like the delay bit and the 1 bit loss signal (or the two bits loss signal).

9. Security Considerations

The privacy considerations for the hybrid RTT measurement signal are essentially the same as those for passive RTT measurement in general.

10. Acknowledgements

tbc

11. IANA Considerations

tbc

12. References

12.1. Normative References

[I-D.ietf-quic-spin-exp]
Trammell, B. and M. Kuehlewind, "The QUIC Latency Spin Bit", draft-ietf-quic-spin-exp-01 (work in progress), October 2018.

- [I-D.ietf-quic-transport]
Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", draft-ietf-quic-transport-23 (work in progress), September 2019.
- [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>>.
- [RFC7799] Morton, A., "Active and Passive Metrics and Methods (with Hybrid Types In-Between)", RFC 7799, DOI 10.17487/RFC7799, May 2016, <<https://www.rfc-editor.org/info/rfc7799>>.
- [RFC8321] Fioccola, G., Ed., Capello, A., Cociglio, M., Castaldelli, L., Chen, M., Zheng, L., Mirsky, G., and T. Mizrahi, "Alternate-Marking Method for Passive and Hybrid Performance Monitoring", RFC 8321, DOI 10.17487/RFC8321, January 2018, <<https://www.rfc-editor.org/info/rfc8321>>.

12.2. Informative References

- [I-D.trammell-ippm-spin]
Trammell, B., "An Explicit Transport-Layer Signal for Hybrid RTT Measurement", draft-trammell-ippm-spin-00 (work in progress), January 2019.
- [I-D.trammell-quic-spin]
Trammell, B., Vaere, P., Even, R., Fioccola, G., Fossati, T., Ihlar, M., Morton, A., and S. Emile, "Adding Explicit Passive Measurability of Two-Way Latency to the QUIC Transport Protocol", draft-trammell-quic-spin-03 (work in progress), May 2018.
- [I-D.trammell-tsvwg-spin]
Trammell, B., "A Transport-Independent Explicit Signal for Hybrid RTT Measurement", draft-trammell-tsvwg-spin-00 (work in progress), July 2018.

Authors' Addresses

Mauro Cociglio
Telecom Italia
Via Reiss Romoli, 274
Torino 10148
Italy

Email: mauro.cociglio@telecomitalia.it

Giuseppe Fioccola
Huawei Technologies
Riesstrasse, 25
Munich 80992
Germany

Email: giuseppe.fioccola@huawei.com

Fabio Bulgarella
Politecnico di Torino

Email: fabio.bulgarella@guest.telecomitalia.it

Riccardo Sisto
Politecnico di Torino
Corso Duca degli Abruzzi, 24
Torino 10129
Italy

Email: riccardo.sisto@polito.it

RTGWG Working Group
INTERNET-DRAFT
Intended Status: Informational
Expires: May 7, 2020

L. Geng
China Mobile
P. Willis
BT
November 4, 2019

Compute First Networking (CFN) Scenarios and Requirements
draft-geng-rtgwg-cfn-req-00

Abstract

Service providers are exploring the edge computing to achieve better response times and transfer rate by moving the computing towards the edge of the network in scenarios like 5G MEC, virtualized central office, etc. Providing services by sharing computing resources from multiple edges is emerging. The service nodes from multiple edges normally have two key features, service equivalency and service dynamics. When the computing resources attached to a single edge site is overloaded, static service dispatch can possibly keep allocating the service request to it and cause inefficient utilization. The service request to edge computing needs to be dispatched to and served by the most suitable edge to improve user experience and system utilization by taking both the available computing resources and network conditions into account.

This draft describes scenarios and requirements of Compute First Networking (CFN) to make the computing and network resource to be considered in a collaborative way to achieve a more balanced network-based distributed service dispatching.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

Copyright and License Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1 Terminology	4
2. Usage Scenarios of CFN	4
2.1 Cloud Based Recognition in Augmented Reality (AR)	4
2.2 Connected Car	5
2.3 Cloud Virtual Reality (VR)	5
3. Requirements	5
4. Summary	6
5. Security Considerations	7
6. IANA Considerations	7
7. Acknowledgements	7
8. References	7
8.1 Normative References	7

1. Introduction

Edge computing aims to provide better response times and transfer rate by moving the computing towards the edge of the network. Edge computing can be built on industrial PCs, embedded systems, gateways and others. They are put close to the end user. There is an emerging requirement that multiple edge sites (called edges too in this document) are deployed at different locations to provide the service. There are millions of home gateways, thousands of base stations and hundreds of central offices in a city that can serve as candidate edges for hosting service nodes. Depending on the location of the edge and its capacity, each edge site may have different computing resources to be used for a particular service. The computing resources hosted on an edge is limited. At peak hour, computing resources attached to the closest edge site may not be sufficient to handle all the incoming requests. Longer response time or even request dropping can be experienced by the user. Increasing the computing resources hosted on each edge site to the potential maximum capacity is neither feasible nor economical.

At the same time, with the new technologies such as serverless computing and container based virtual functions, service node on an edge can be easily created and terminated in a sub-second scale. It makes the available computing resources for a service change dramatically over time at an edge.

The traditional method of static pre-configuration of which service request is dispatched to which edge causes the workload distribution to be unbalanced in terms of network load and computational load. The reason is there is no interaction on scheduling capability between edges about the hosted computing nodes. When computing resources on one edge are overloaded or even unavailable, the requests may still keep coming and cause the service experience deteriorates. Most current solutions use the application layer functions to solve this issue. It requires L4-L7 handling of the packet processing, such as reverse proxy, which takes longer connection time. It is not an efficient approach for huge number of short connections.

Multi-site edge computing has the distributed nature. Service providers are starting to build the edge platform to allow a large number of requests to be served by sharing the computing resources from service nodes at multiple edges in a collaborative way. That is to say, a service request potentially can be handled by different service nodes located in different edges and it has to be decided which one is the most appropriate to serve this request in real time. Such an approach can improve the system utilization to serve more end users by balancing the workload distributed to multiple edges intelligently. Intelligence here means considering both the network

conditions and available computing resources.

Both computing load and network status are treated as network visible resources, edge site can interact with each other to provide network-based service dispatching to achieve better load balancing. This is called Compute First Networking (CFN) in this document. It requires both network, edge and computing nodes to work coordinately for service selection dispatching between user to edge and edge to edge. Among them, edge to edge or inter-edge interaction is the focus of CFN in IETF related work. This draft describes usage scenarios and requirements of CFN.

1.1 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Usage Scenarios of CFN

This section presents several typical scenarios which require multiple edge sites to interconnect and to co-ordinate with networks to meet the service requirements and ensure user experience.

2.1 Cloud Based Recognition in Augmented Reality (AR)

In AR environment, the end device captures the images via cameras and sends out the computing intensive service request. Normally service nodes at the edge are responsible for tasks with medium computational complexity or low latency requirement like object detection, feature extraction and template matching, and service nodes at cloud are responsible for the most intensive computational tasks like object recognition or latency non-sensitive tasks like AI based model training. The end device hence only handles the tasks like target tracking and image display, thereby reducing the computing load of the client.

The computing resource for a specific service at the edge can be instantiated on-demand. Once the task is completed, this resource can be released. The lifetime of such "function as a service" can be on a millisecond scale. Therefore computing resources on the edges have distributed and dynamic natures. A service request has to be sent to and served by an edge with sufficient computing resource and a good

network path.

2.2 Connected Car

In auxiliary driving scenarios, to help overcome the non-line-of-sight problem due to blind spot or obstacles, the edge node can collect the comprehensive road and traffic information around the vehicle location and perform data processing, and then the vehicles in high security risk can be signaled. It improves the driving safety in complicated road conditions, like at the intersections. The video image information captured by the surveillance camera is transmitted to the nearest edge node for processing. Warnings can be sent to the cars driving too fast or under other invisible dangers.

When the local edge node is overloaded, the service request sent to it will be queued and the response from the auxiliary driving will be delayed, and it may lead to traffic accidents. Hence, in such cases, delay-insensitive services such as in-vehicle entertainment should be dispatched to other light loaded nodes instead of local edge nodes, so that the delay-sensitive service is preferentially processed locally to ensure the service availability and user experience.

2.3 Cloud Virtual Reality (VR)

Cloud VR introduces the concept and technology of cloud computing and cloud rendering into VR applications. The end device usually only uploads the posture or control information to the cloud and then VR contents are rendered in the cloud. The video and audio outputs generated from the cloud are encoded, compressed, and transmitted back to the end device via high bandwidth network.

Cloud VR services have high requirements on both network and computing. For example, for an entry-level Cloud VR (panoramic 8K 2D video) with 110-degree Field of View (FOV) transmission, the typical network requirements are bandwidth 40Mbps, RTT 20ms, packet loss rate is $2.4E-5$; the typical computing requirements are 8K H.265 real-time decoding, 2K H.264 real-time encoding.

Cloud VR service brings challenging requirements on both network and computing so that the edge node to serve the request has to be carefully selected to avoid the overloading.

3. Requirements

CFN in this document mainly targets at the typical edge computing

scenarios with two key features, service equivalency and service dynamics.

- Service equivalency: Equivalent service is provided by multiple edges to ensure better scalability and availability.
- Service dynamics: A single edge has very dynamic resources over time to serve a request. Its dynamics are affected by computing resource of service node, network path congestion, failover and others.

A service request should be routed to the most suitable edge for processing. The local edge is normally the first choice. At the same time, it is important to have the capability to route the request to the other edges when the local edge has insufficient computing resource or non-promising network path, depending on the service type and/or priority.

The following requirements need to be met for CFN,

- The service provided, or function called, should be identified in a format amenable to processing in the network
- Service request is sent in real time to the most appropriate one among all the service equivalent edges for processing. Such a request assignment should not be static. It should be based on the metrics for the service dynamics, including both network status and available computing resources.
- For applications that require flow affinity it must be possible to have a method to signal flow affinity requirements and handle flows on the same edge.
- Edge nodes may have limited compute resources therefore control and storage overhead must be minimised

4. Summary

CFN in this document tries to leverage the network distributed nature to help serve the edge computing requests in a more balanced way by considering both network status and computing resources among multiple edges. Inter-edge interaction is required in the dynamic service dispatching and network and computing resource information distribution.

This document illustrate some usage scenarios and requirements for CFN. CFN architecture should addresses how to distribute the computing resource information at the network layer, how the data

plane adapts when the edge to handle the first service request is not known in advance, and how to assure flow affinity.

5. Security Considerations

TBD

6. IANA Considerations

No IANA action is required.

7. Acknowledgements

The author would like to thank all participants who participated in the discussion of CFN at the earlier IETF/IRTF meetings.

8. References

8.1 Normative References

Authors' Addresses

Liang Geng
China Mobile
Email: gengliang@chinamobile.com

Peter Willis
BT
Email: peter.j.willis@bt.com

RTGWG
INTERNET-DRAFT
Intended Status: Informational

S. Gu
G. Zhuang
Huawei Technologies
H. Yao
X. Li
China Mobile

Expires: June 4, 2020

December 2, 2019

A Report on Compute First Networking (CFN) Field Trial
draft-gu-rtgwg-cfn-field-trial-01

Abstract

Compute First Networking (CFN) enables the routing of the service request to an optimal edge site to improve the overall system load balancing and efficiency. Especially when an edge site is overloaded, other edges with service equivalency can dynamically serve the request. This document describes a CFN field trial to show the effect that CFN can achieve. Edge to edge interaction to get the available computing resources information for services and the network status to each other is introduced. Data plane to support late binding based dynamic anycast is illustrated too. The field trial shows that CFN can greatly improve the overall query per second served for a service hosted on multiple edges in a more balanced way.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at
<http://www.ietf.org/shadow.html>

Copyright and License Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1 Terminology	3
2 Testbed overview	3
3. Procedures	5
3.1 Control Plane	5
3.2 Data Plane	6
4. Preliminary Tests	9
4.1 Requests rush to an edge (no system background load)	9
4.2 Requests rush to an edge (system background load exists) . .	10
4.3 Mixed requests rush to an edge (no system background load) .	11
4.4 Impact from update frequency	12
5. Summary	13
6. Security Considerations	13
7. IANA Considerations	13
8. Acknowledgements	13
9. References	13
9.1 Normative References	13
9.2 Informative References	14
Authors' Addresses	14

1. Introduction

Compute First Networking (CFN) Scenarios and Requirements [CFN-req] shows the usage scenarios and requirements to dynamically dispatch the service request to multiple edge sites in order to overcome the computing resource overloading problem in edge computing. Compute First Networking (CFN) framework document [CFN-fmwk] presents the basic system framework to dynamically route a service request to a selected edge in real time based on the computing load status and network conditions. This approach improves the load balancing between multiple edges with service equivalency in a distributed manner. This document introduces a more concrete CFN field trial and its performance.

1.1 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2 Testbed overview

We deployed CFN node on three edge sites in Hangzhou. The sites are approximately 30 kilometers apart. Figure 1 shows the topology and configuration we used for this CFN testbed.

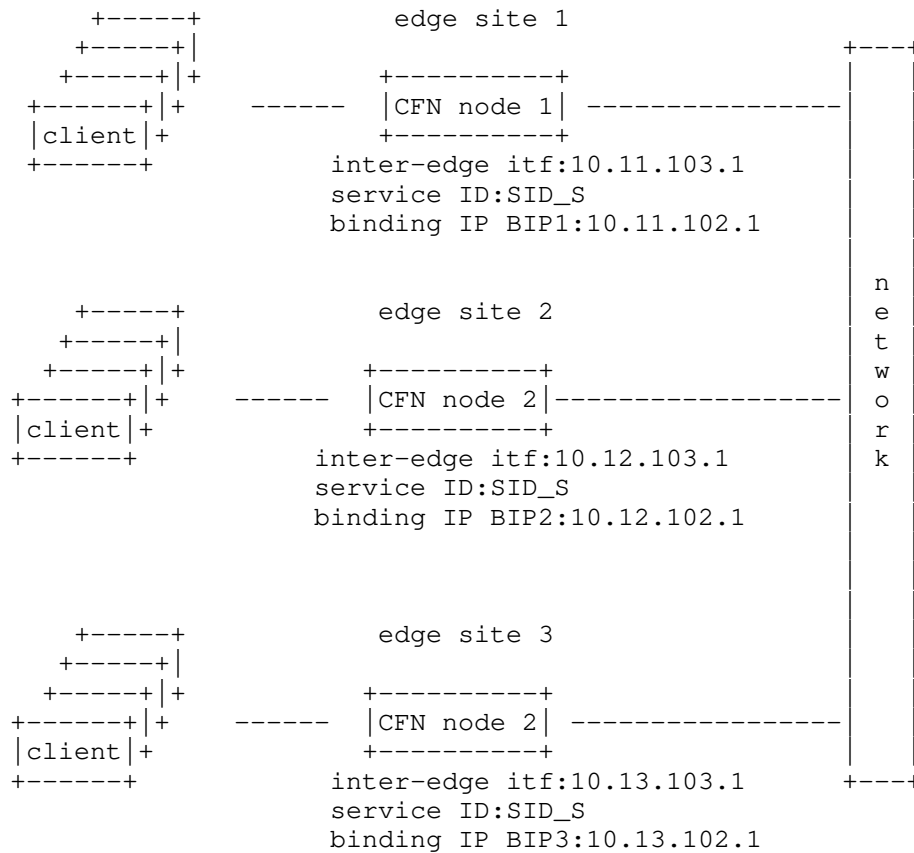


Figure 1. CFN testbed overview

A matrix multiplication service *S* is provided by all three edge sites (or edges for simplicity in this document). The CFN nodes use a unique service ID *SID_S* to announce the its reachability to service *S*. In our test, we use 200.200.200.201 for *SID_S*. Consider *SID_S* here as a anycast IP address. Though this service is reachable by a single *SID_S* in network, 3 edges indeed serve *SID_S* using 3 different binding IP (BIP) addresses , BIP1/2/3 with address 10.11/12/13.102.1 via CFN node 1/2/3 respectively. Service node hosted on or attached to a CFN node only knows that it uses its BIP to serve service *S* and has no knowledge about *SID_S*.

Each CFN node has an inter-edge interface IP address for communicating the computing load information among CFN nodes. About 200 simulated clients connect to each CFN node in the test.

3. Procedures

The procedures are introduced in [CFN-fmwk]. For easy reference, control plane and data plane timeline diagrams are shown here too.

3.1 Control Plane

When a service node is initiated for service *S*, the edge platform manager will send the registration information about service ID *SID_S* and binding IP (BIP) to access *SID_S* to the CFN node that the service node attaches to.

Each CFN node regularly gets the computing load information about the service node attached to it for *SID_S*. The computing load information can be CPU consumption for *SID_S*, number of current connections, query per second processed, total capacity, or other performance metrics. In our test, we give each type of metrics a weight. CFN nodes distribute those information to each other by BGP extensions. Figure 2 shows the CFN control plane procedures.

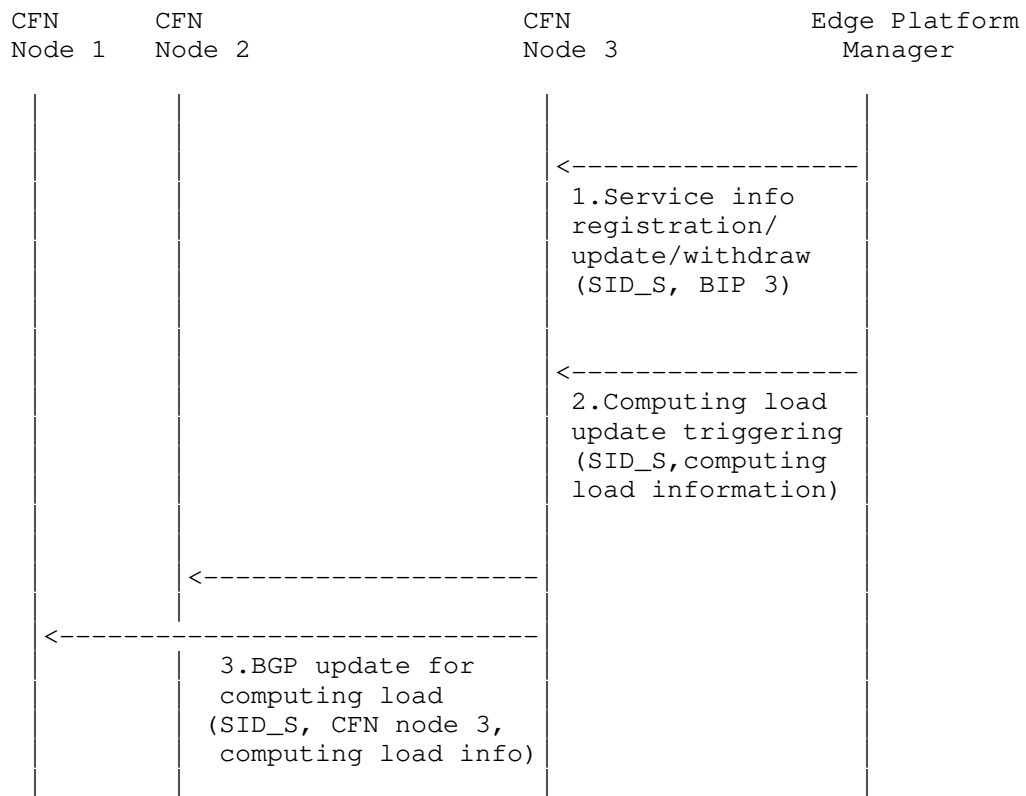


Figure 2. CFN control plane

3.2 Data Plane

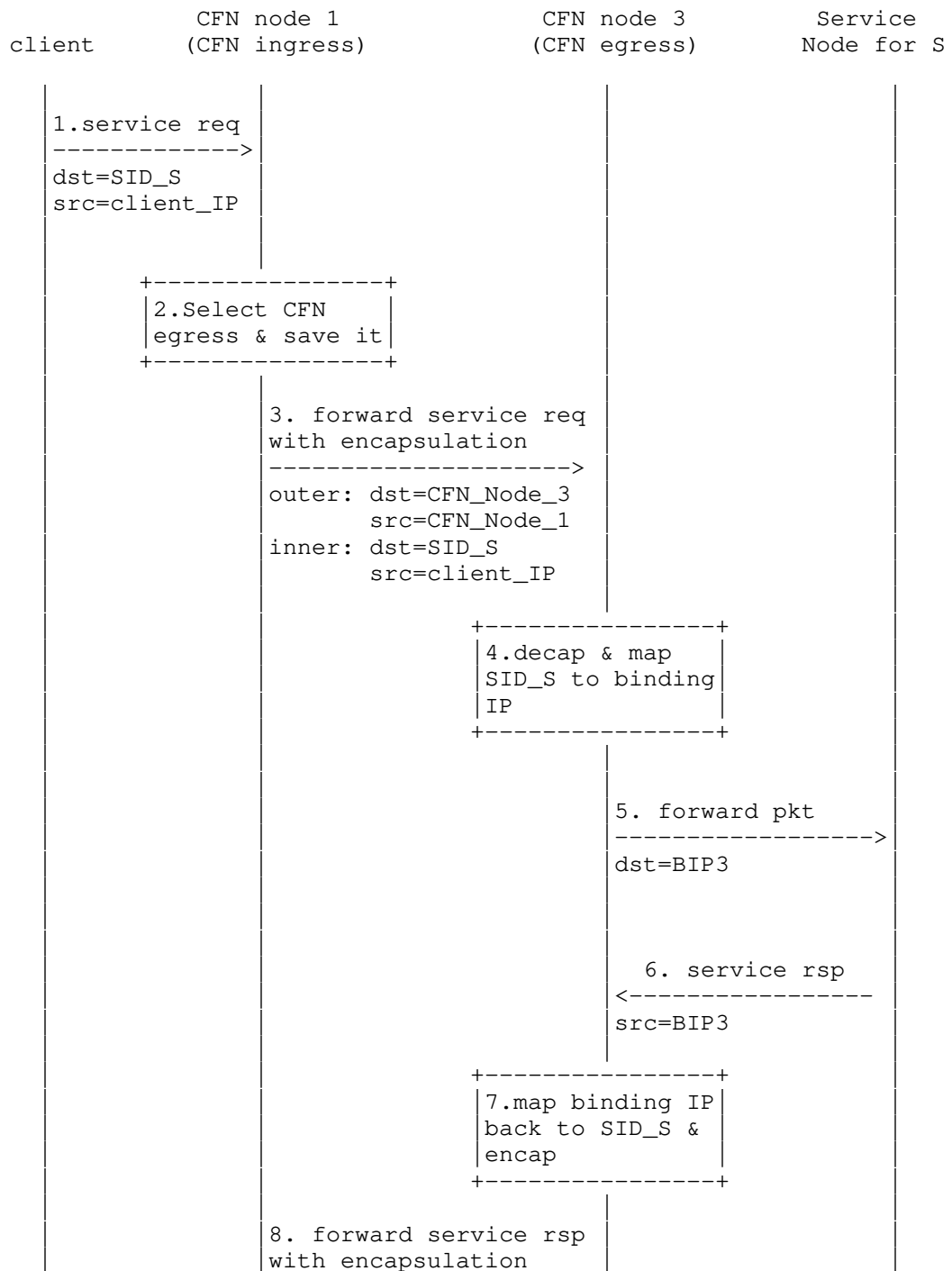
When a client sends a service request for service S, it uses SID_S as destination IP. In the test, SID_S is an anycast address. There are various ways that a client can get the SID_S for a service, such as by DNS or static configuration.

When the CFN ingress which is CFN node 1 in figure 3 receives the request, it dynamically selects the most appropriate CFN egress based on computing load information received. As figure 4 shows, CFN node 3 is selected as CFN egress in this case. CFN ingress further tunnels the data packet to CFN egress.

When CFN egress receives the packet, it decapsulates the packet and maps the destination address from SID_S to binding IP BIP3. The service node for service S gets the packet and processes it. The

service response is returned back to CFN node 3. CFN node 3 is conceptually the gateway of attached service nodes for CFN services. It maps BIP3 to SID_S as source IP and then tunnels it to CFN node 1. CFN node 1 further decapsulates the packet and sends it to the client.

For the subsequent service request packets sent to CFN node 1 from the same flow, CFN node always uses CFN node 3 as the egress to ensure the flow affinity.



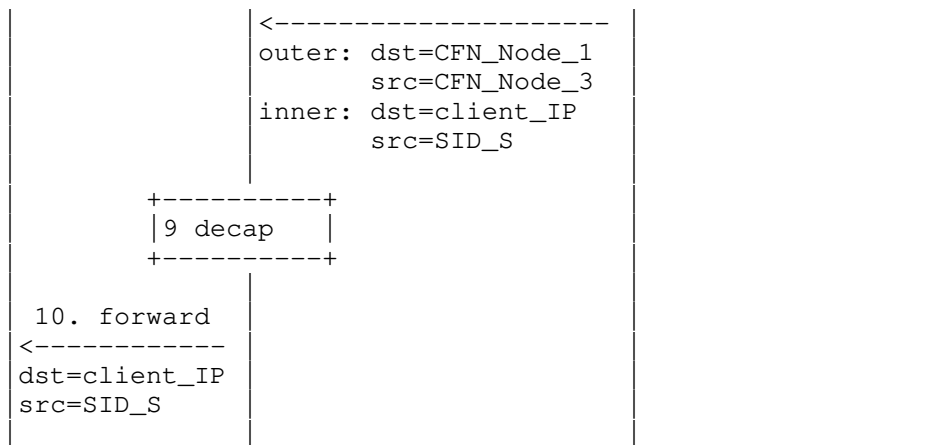


Figure 3. CFN data plane for the first request of a flow

4. Preliminary Tests

4.1 Requests rush to an edge (no system background load)

In this test, we assume the service nodes capacities attached to all three edges are the same and there is no background computing tasks running. The overall computing task handling capacity from service nodes can handle about 670 queries per second (qps).

The clients attached to edge 1 generating service request to it at about 40 qps. The number of clients simultaneously send requests varies. When 10 clients send requests, the computing power consumed by the system can reach approximately 60% of its overall maximum. The requests are all short-processing tasks and based on observation each request roughly take 4ms to be completed at the server side.

CFN leverages the computing load reported by different edges and together with network status to spread the service request. On the other hand, a pure random selection from the edges to handle the request is used for comparison.

We tested for 5, 10 and 15 clients attached to one edge which result in the consumption of medium low, medium high and high computing resources of the whole system respectively. Note it exceeds a single edge capacity in any case. For 15 clients case, it almost reaches the maximum system capacity. Figure 4 shows the average delay between a request being sent and the response being received by a client and

system qps.

number of clients	system	average delay (ms)	qps
5 (medium low)	CFN	3.954	208.5
	random	5.316	197.7
10 (medium high)	CFN	4.700	402.3
	random	5.595	302.1
15 (high)	CFN	5.506	559.3
	random	5.718	546.0

Figure 4. Test results when service requests rush to a single edge when no system background load

The CFN achieves better results compared with random selection based application layer service dispatch. Average delay decreased by 25.62% and 16.00% and total qps increased by 5.5% and 33.17% in medium low and medium high computing load respectively. The unbalanced incoming traffic is spread to all edges. Unlike random selection, CFN will dispatch more requests to the local edge since its network cost is the lowest. CFN balances between higher computing resources available at the remote sites and lower network cost at the local site to make a choice. Hence it outperforms the random selection. In high number of clients case, as the maximum system capacity is almost reached, the performance are similar for CFN and random case.

4.2 Requests rush to an edge (system background load exists)

In this test, different edge has different background computing tasks to handle. We randomly select an edge to make it suffer from a computing intensive burst which consumes almost 90% of its capacity for about 4 seconds. Then computing load returns to zero for 2 seconds. It creates the busy edge and idle edges scenario. The other settings are same as shown in section 4.1.

Figure 5 shows the average delay between a request being sent and the

response being received by a client and system qps for this case.

number of clients	system	average delay (ms)	qps
5 (medium low)	CFN	6.291	185.6
	random	9.630	165.3
10 (medium high)	CFN	6.854	360.9
	random	10.592	316.3
15 (high)	CFN	7.987	512.4
	random	12.156	441.7

Figure 5. Test results when service requests rush to a single edge when system background load exists

The results show that CFN has average delay decreased by 34.67%, 35.29% and 34.30% in medium low, medium high and high computing load respectively. And total qps is increased by 12.28%, 14.10% and 16.01% in medium low, medium high and high computing load respectively.

The performance gain of CFN shown in this test case is much higher than that in section 4.1 The reason is that the random service dispatching has more than 20% chance to send the request to an edge with service node with very high background computing load while CFN can greatly reduce such possibility.

In addition, compare with the results in section 4.1, delay increases 59.10%, 45.83% and 45.06% in different computing load level in CFN and 81.15%, 89.31%, 112.60% in random selection. It shows CFN can much better adapt to dynamic computing load change especially when system background load is high.

4.3 Mixed requests rush to an edge (no system background load)

We changed the characteristics of service requests to reflect the co-existence nature of long-processing tasks and short-processing tasks. Short-processing task takes roughly 4ms to complete and long-processing task takes roughly 400ms to complete. And the ratio of

long and short tasks is approximately 1:100.

Figure 6 shows the average delay between a request being sent and the response being received by a client and system qps for this case.

number of clients	system	average delay (ms)	qps
5 (medium low)	CFN	5.205	193.5
	random	5.398	193.5
10 (medium high)	CFN	5.201	393.4
	random	5.985	385
15 (high)	CFN	6.147	559.4
	random	8.499	559.4

Figure 6. Test results when mixed service requests rush to a single edge when no system background load

The results show that CFN has average delay decreased by 3.58%, 13.10% and 27.76% in medium low, medium high and high computing load respectively. The qps has no much difference for different levels of computing load especially for the medium low and high case.

4.4 Impact from update frequency

The computing load information is updated and distributed when its metric changes exceed some threshold compared to the last distributed information. In the test, we used the 10% of maximum number of connections allowed and 5% CPU consumption as threshold. Frequency of update affects the system performance. We tested for different update interval to see their impact. The clients keep sending requests to make the computing resource consumption on each edge maintained at medium low which is about 5 connections. Update interval has been set to 10s, 5s, 1s, 100ms, 10ms, 1ms. Figure 7 shows the average delay between a request being sent and the response being received by a client under different update intervals and the improvement of delay when comparing to the case of 10 second interval.

The results shows that the higher frequency of updates distributed the better performance.

# of clients	Interval	10s	5s	1s	100ms	10ms	1ms
5 (medium low)	Delay (us)	6445	6255	5741	5312	4883	4058
	Improvement (%)	0	3.5	12.3	21.3	32.3	58.8

Figure 7. Test results under different update intervals

5. Summary

This draft presents a field trial for CFN system with three edge sites in different locations. CFN enables a network-based fast-react system to serve multi-edge based computing service in a more balanced way. Computing load information are exchanged regularly between CFN nodes. CFN egress bound to serve a particular service is determined in real time and maintained to ensure flow affinity.

The tests show that the overall clients' request delay is greatly decreased and the system qps has some improvement too. CFN is a feasible and efficient way in edge computing to provide multi-edge service balancing.

6. Security Considerations

The security risks mentioned in [CFN-fmwk] apply in the tests. As a preliminary tests, no extra security risks control is implemented currently. Mechanisms such as authentication of edge node and fluctuation avoidance should be considered in deployment.

7. IANA Considerations

No IANA action is required.

8. Acknowledgements

The authors would like to thank Xunwen Li's team members for their help in setting up the testbed in Hangzhou.

9. References

9.1 Normative References

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

9.2 Informative References

[CFN-req] Geng, L., et al, "Compute First Networking (CFN) Scenarios and Requirements", draft-geng-cfn-req-00, November 2019.

[CFN-fmwk] Li, Y., et al, "Framework of Compute First Networking (CFN)", draft-li-cfn-framework-00, November 2019.

Authors' Addresses

Shuheng Gu
Huawei Technologies

EMail: gushuheng@huawei.com

Guanhua Zhuang
Huawei Technologies

EMail: zhuangguanhua@huawei.com

Huijuan Yao
China Mobile

EMail: yaohuijuan@chinamobile.com

Xunwen Li
China Mobile

EMail: lixunwen@zj.chinamobile.com

Mboned
Internet-Draft
Updates: 7450 (if approved)
Intended status: Standards Track
Expires: April 29, 2020

J. Holland
Akamai Technologies, Inc.
October 27, 2019

DNS Reverse IP AMT Discovery
draft-ietf-mboned-driad-amt-discovery-09

Abstract

This document updates RFC 7450 (Automatic Multicast Tunneling, or AMT) by extending the relay discovery process to use a new DNS resource record named AMTRELAY when discovering AMT relays for source-specific multicast channels. The reverse IP DNS zone for a multicast sender's IP address is configured to use AMTRELAY resource records to advertise a set of AMT relays that can receive and forward multicast traffic from that sender over an AMT tunnel.

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 <https://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 April 29, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

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

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Background	4
1.2. Terminology	4
1.2.1. Relays and Gateways	4
1.2.2. Definitions	4
2. Relay Discovery Operation	5
2.1. Overview	6
2.2. Signaling and Discovery	6
2.3. Optimal Relay Selection	8
2.3.1. Overview	8
2.3.2. Preference Ordering	10
2.3.3. Connecting to Multiple Relays	12
2.4. Happy Eyeballs	12
2.4.1. Overview	12
2.4.2. Algorithm Guidelines	13
2.4.3. Connection Definition	14
2.5. Guidelines for Restarting Discovery	14
2.5.1. Overview	14
2.5.2. Updates to Restarting Events	15
2.5.3. Tunnel Stability	16
2.5.4. Traffic Health	16
2.5.5. Relay Loaded or Shutting Down	18
2.5.6. Relay Discovery Messages vs. Restarting Discovery	18
2.5.7. Independent Discovery Per Traffic Source	19
2.6. DNS Configuration	19
2.7. Waiting for DNS resolution	20
3. Example Deployments	20
3.1. Example Receiving Networks	20
3.1.1. Tier 3 ISP	20
3.1.2. Small Office	21
3.2. Example Sending Networks	23
3.2.1. Sender-controlled Relays	23
3.2.2. Provider-controlled Relays	24
4. AMTRELAY Resource Record Definition	25
4.1. AMTRELAY RRTYPE	25
4.2. AMTRELAY RData Format	25
4.2.1. RData Format - Precedence	26
4.2.2. RData Format - Discovery Optional (D-bit)	26
4.2.3. RData Format - Type	26
4.2.4. RData Format - Relay	27
4.3. AMTRELAY Record Presentation Format	27
4.3.1. Representation of AMTRELAY RRs	27

4.3.2. Examples	28
5. IANA Considerations	28
6. Security Considerations	29
6.1. Use of AMT	29
6.2. Record-spoofing	29
6.3. Congestion	30
7. Acknowledgements	30
8. References	30
8.1. Normative References	30
8.2. Informative References	32
Appendix A. Unknown RRTYPE construction	33
Author's Address	34

1. Introduction

This document defines DNS Reverse IP AMT Discovery (DRIAD), a mechanism for AMT gateways to discover AMT relays that are capable of forwarding multicast traffic from a known source IP address.

AMT (Automatic Multicast Tunneling) is defined in [RFC7450], and provides a method to transport multicast traffic over a unicast tunnel, in order to traverse non-multicast-capable network segments.

Section 4.1.5 of [RFC7450] explains that the relay selection process for AMT is intended to be more flexible than the particular discovery method described in that document, and further explains that the selection process might need to depend on the source of the multicast traffic in some deployments, since a relay must be able to receive multicast traffic from the desired source in order to forward it.

That section goes on to suggest DNS-based queries as a possible solution. DRIAD is a DNS-based solution, as suggested there. This solution also addresses the relay discovery issues in the "Disadvantages" lists in Section 3.3 of [RFC8313] and Section 3.4 of [RFC8313].

The goal for DRIAD is to enable multicast connectivity between separate multicast-enabled networks when neither the sending nor the receiving network is connected to a multicast-enabled backbone, without pre-configuring any peering arrangement between the networks.

This document updates Section 5.2.3.4 of [RFC7450] by adding a new extension to the relay discovery procedure.

1.1. Background

The reader is assumed to be familiar with the basic DNS concepts described in [RFC1034], [RFC1035], and the subsequent documents that update them, particularly [RFC2181].

The reader is also assumed to be familiar with the concepts and terminology regarding source-specific multicast as described in [RFC4607] and the use of IGMPv3 [RFC3376] and MLDv2 [RFC3810] for group management of source-specific multicast channels, as described in [RFC4604].

The reader should also be familiar with AMT, particularly the terminology listed in Section 3.2 of [RFC7450] and Section 3.3 of [RFC7450].

1.2. Terminology

1.2.1. Relays and Gateways

When reading this document, it's especially helpful to recall that once an AMT tunnel is established, the relay receives native multicast traffic and sends unicast tunnel-encapsulated traffic to the gateway, and the gateway receives the tunnel-encapsulated packets, decapsulates them, and forwards them as native multicast packets, as illustrated in Figure 1.

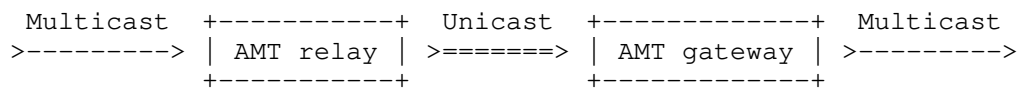


Figure 1: AMT Tunnel Illustration

1.2.2. Definitions

Term	Definition
(S,G)	A source-specific multicast channel, as described in [RFC4607]. A pair of IP addresses with a source host IP and destination group IP.
discovery broker	A broker or load balancer for AMT relay discovery, as mentioned in section 4.2.1.1 of [RFC7450].
downstream	Further from the source of traffic, as described in [RFC7450].
FQDN	Fully Qualified Domain Name, as described in [RFC8499]
gateway	An AMT gateway, as described in [RFC7450]
L flag	The "Limit" flag described in Section 5.1.1.4 of [RFC7450]
relay	An AMT relay, as described in [RFC7450]
RPF	Reverse Path Forwarding, as described in [RFC5110]
RR	A DNS Resource Record, as described in [RFC1034]
RRType	A DNS Resource Record Type, as described in [RFC1034]
SSM	Source-specific multicast, as described in [RFC4607]
upstream	Closer to the source of traffic, as described in [RFC7450].

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] and [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Relay Discovery Operation

2.1. Overview

The AMTRELAY resource record (RR) defined in this document is used to publish the IP address or domain name of a set of AMT relays or discovery brokers that can receive, encapsulate, and forward multicast traffic from a particular sender.

The sender is the owner of the RR, and configures the zone so that it contains a set of RRs that provide the addresses or domain names of AMT relays (or discovery brokers that advertise relays) that can receive multicast IP traffic from that sender.

This enables AMT gateways in remote networks to discover an AMT relay that is capable of forwarding traffic from the sender. This in turn enables those AMT gateways to receive the multicast traffic tunneled over a unicast AMT tunnel from those relays, and then to pass the multicast packets into networks or applications that are using the gateway to subscribe to traffic from that sender.

This mechanism only works for source-specific multicast (SSM) channels. The source address of the (S,G) is reversed and used as an index into one of the reverse mapping trees (in-addr.arpa for IPv4, as described in Section 3.5 of [RFC1035], or ip6.arpa for IPv6, as described in Section 2.5 of [RFC3596]).

This mechanism should be treated as an extension of the AMT relay discovery procedure described in Section 5.2.3.4 of [RFC7450]. A gateway that supports this method of AMT relay discovery SHOULD use this method whenever it's performing the relay discovery procedure, and the source IP addresses for desired (S,G)s are known to the gateway, and conditions match the requirements outlined in Section 2.3.

Some detailed example use cases are provided in Section 3, and other applicable example topologies appear in Section 3.3 of [RFC8313], Section 3.4 of [RFC8313], and Section 3.5 of [RFC8313].

2.2. Signaling and Discovery

This section describes a typical example of the end-to-end process for signaling a receiver's join of an SSM channel that relies on an AMTRELAY RR.

The example in Figure 2 contains 2 multicast-enabled networks that are both connected to the internet with non-multicast-capable links, and which have no direct association with each other.

A content provider operates a sender, which is a source of multicast traffic inside a multicast-capable network.

An end user who is a customer of the content provider has a multicast-capable internet service provider, which operates a receiving network that uses an AMT gateway. The AMT gateway is DRIAD-capable.

The content provider provides the user with a receiving application that tries to subscribe to at least one (S,G). This receiving application could for example be a file transfer system using FLUTE [RFC6726] or a live video stream using RTP [RFC3550], or any other application that might subscribe to an SSM channel.

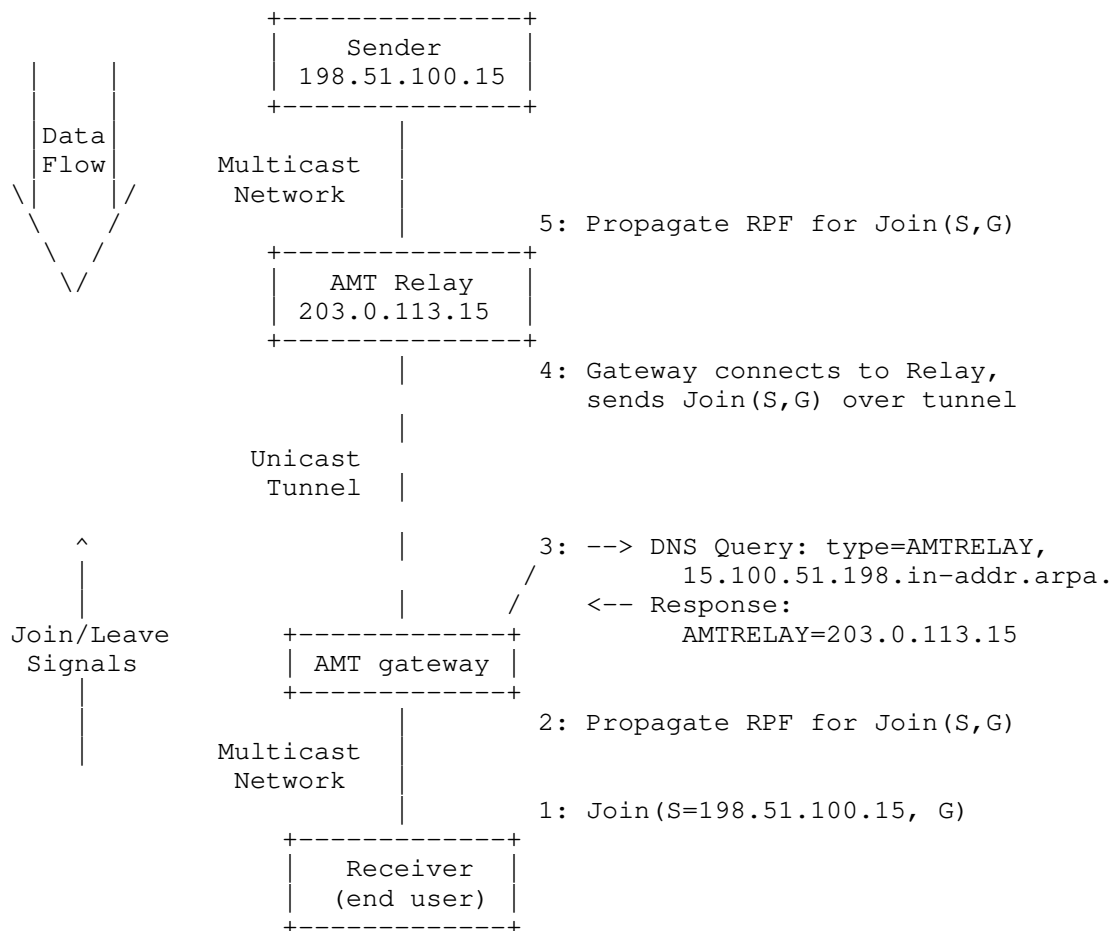


Figure 2: DRIAD Messaging

In this simple example, the sender IP is 198.51.100.15, and the relay IP is 203.0.113.15.

The content provider has previously configured the DNS zone that contains the domain name "15.100.51.198.in-addr.arpa.", which is the reverse lookup domain name for his sender. The zone file contains an AMTRELAY RR with the Relay's IP address. (See Section 4.3 for details about the AMTRELAY RR format and semantics.)

The sequence of events depicted in Figure 2 is as follows:

1. The end user starts the app, which issues a join to the (S,G): (198.51.100.15, 232.252.0.2).
2. The join propagates with RPF through the receiver's multicast-enabled network with PIM [RFC7761] or another multicast routing mechanism, until the AMT gateway receives a signal to join the (S,G).
3. The AMT gateway performs a reverse DNS lookup for the AMTRELAY RRTYPE, by sending an AMTRELAY RRTYPE query for the FQDN "15.100.51.198.in-addr.arpa.", using the reverse IP domain name for the sender's source IP address (the S from the (S,G)), as described in Section 3.5 of [RFC1035].

The DNS resolver for the AMT gateway uses ordinary DNS recursive resolution until it has the authoritative result that the content provider configured, which informs the AMT gateway that the relay address is 203.0.113.15.

4. The AMT gateway performs AMT handshakes with the AMT relay as described in Section 4 of [RFC7450], then forwards a Membership report to the relay indicating subscription to the (S,G).
5. The relay propagates the join through its network toward the sender, then forwards the appropriate AMT-encapsulated traffic to the gateway, which decapsulates and forwards it as native multicast through its downstream network to the end user.

2.3. Optimal Relay Selection

2.3.1. Overview

The reverse source IP DNS query of an AMTRELAY RR is a good way for a gateway to discover a relay that is known to the sender.

However, it is NOT necessarily a good way to discover the best relay for that gateway to use, because the RR will only provide information about relays known to the source.

If there is an upstream relay in a network that is topologically closer to the gateway and able to receive and forward multicast traffic from the sender, that relay is better for the gateway to use, since more of the network path uses native multicast, allowing more chances for packet replication. But since that relay is not known to the sender, it won't be advertised in the sender's reverse IP DNS record. An example network that illustrates this scenario is outlined in Section 3.1.2.

It's only appropriate for an AMT gateway to discover an AMT relay by querying an AMTRELAY RR owned by a sender when all of these conditions are met:

1. The gateway needs to propagate a join of an (S,G) over AMT, because in the gateway's network, no RPF next hop toward the source can propagate a native multicast join of the (S,G); and
2. The gateway is not already connected to a relay that forwards multicast traffic from the source of the (S,G); and
3. The gateway is not configured to use a particular IP address for AMT discovery, or a relay discovered with that IP is not able to forward traffic from the source of the (S,G); and
4. The gateway is not able to find an upstream AMT relay with DNS-SD [RFC6763], using "_amt._udp" as the Service section of the queries, or a relay discovered this way is not able to forward traffic from the source of the (S,G) (as described in Section 2.5.4.1 or Section 2.5.5); and
5. The gateway is not able to find an upstream AMT relay with the well-known anycast addresses from Section 7 of [RFC7450].

When the above conditions are met, the gateway has no path within its local network that can receive multicast traffic from the source IP of the (S,G).

In this situation, the best way to find a relay that can forward the required traffic is to use information that comes from the operator of the sender. When the sender has configured an AMTRELAY RR, gateways can use the DRIAD mechanism defined in this document to discover the relay information provided by the sender.

2.3.2. Preference Ordering

This section defines a preference ordering for relay addresses during the relay discovery process. Gateways are encouraged to implement a Happy Eyeballs algorithm to try candidate relays concurrently, but even gateways that do not implement a Happy Eyeballs algorithm SHOULD use this ordering, except as noted.

When establishing an AMT tunnel to forward multicast data, it's very important for the discovery process to prioritize the network topology considerations ahead of address selection considerations, in order to gain the packet replication benefits from using multicast instead of unicast tunneling in the multicast-capable portions of the network path.

The intent of the advice and requirements in this section is to describe how a gateway should make use of the concurrency provided by a Happy Eyeballs algorithm to reduce the join latency, while still prioritizing network efficiency considerations over Address Selection considerations.

Section 4 of [RFC8305] requires a Happy Eyeballs algorithm to sort the addresses with the Destination Address Selection defined in Section 6 of [RFC6724], but for the above reasons, that requirement is superseded in the AMT discovery use case by the following considerations:

- o Prefer Local Relays

Figure 5 and Section 3.1.2 provide a motivating example to prefer DNS-SD [RFC6763] for discovery strictly ahead of using the AMTRELAY RR controlled by the sender for AMT discovery.

For this reason, it's RECOMMENDED that AMT gateways by default perform service discovery using DNS Service Discovery (DNS-SD) [RFC6763] for `_amt._udp.<domain>` (with `<domain>` chosen as described in Section 11 of [RFC6763]) and use the AMT relays discovered that way in preference to AMT relays discoverable via the mechanism defined in this document (DRIAD).

- o Prefer Relays Managed by the Containing Network

When no local relay is discoverable with DNS-SD, it still may be the case that a relay local to the receiver is operated by the network providing transit services to the receiver.

In this case, when the network cannot make the relay discoverable via DNS-SD, the network SHOULD use the well-known anycast

addresses from Section 7 of [RFC7450] to route discovery traffic to the relay most appropriate to the receiver's gateway.

Accordingly, the gateway SHOULD by default discover a relay with the well-known AMT anycast addresses as the second preference after DNS-SD when searching for a local relay.

- o Let Sender Manage Relay Provisioning

A related motivating example in the sending-side network is provided by considering a sender that needs to instruct the gateways on how to select between connecting to Figure 6 or Figure 7 (from Section 3.2), in order to manage load and failover scenarios in a manner that operates well with the sender's provisioning strategy for horizontal scaling of AMT relays.

In this example about the sending-side network, the precedence field described in Section 4.2.1 is a critical method of control so that senders can provide the appropriate guidance to gateways during the discovery process.

Therefore, after DNS-SD, the precedence from the RR MUST be used for sorting preference ahead of the Destination Address Selection ordering from Section 6 of [RFC6724], so that only relay IPs with the same precedence are directly compared according to the Destination Address Selection ordering.

Accordingly, AMT gateways SHOULD by default prefer relays in this order:

1. DNS-SD
2. Anycast addresses from Section 7 of [RFC7450]
3. DRIAD

This default behavior MAY be overridden by administrative configuration where other behavior is more appropriate for the gateway within its network.

Among relay addresses that have an equivalent preference as described above, a Happy Eyeballs algorithm for AMT MUST use the Destination Address Selection defined in Section 6 of [RFC6724], as required by [RFC8305].

Among relay addresses that still have an equivalent preference after the above orderings, a gateway MUST make a non-deterministic choice for relay preference ordering, in order to support load balancing by DNS configurations that provide many relay options.

The gateway MAY introduce a bias in the non-deterministic choice according to information obtained out of band or from a historical record about network topology, timing information, or the response to a probing mechanism, that indicates some expected benefits from selecting some relays in preference to others. Details about the structure and collection of this information are out of scope for this document, but a gateway in possession of such information MAY use it to prefer topologically closer relays.

Note also that certain relay addresses might be excluded from consideration by the hold-down timers described in Section 2.5.4.1 or Section 2.5.5. These relays constitute "unusable destinations" under Rule 1 of the Destination Address Selection, and are also not part of the superseding considerations described above.

The discovery and connection process for the relay addresses in the above described ordering MAY operate in parallel, subject to delays prescribed by the Happy Eyeballs requirements described in Section 5 of [RFC8305] for successively launched concurrent connection attempts.

2.3.3. Connecting to Multiple Relays

In some deployments, it may be useful for a gateway to connect to multiple upstream relays and subscribe to the same traffic, in order to support an active/active failover model. A gateway SHOULD NOT be configured to do so without guaranteeing that adequate bandwidth is available.

A gateway configured to do this SHOULD still use the same preference ordering logic from Section 2.3.2 for each connection. (Note that this ordering allows for overriding by explicit administrative configuration where required.)

2.4. Happy Eyeballs

2.4.1. Overview

Often, multiple choices of relay will exist for a gateway using DRIAD for relay discovery. Happy Eyeballs [RFC8305] provides a widely deployed and generalizable strategy for probing multiple possible connections in parallel, therefore it is RECOMMENDED that DRIAD-capable gateways implement a Happy Eyeballs algorithm to support fast discovery of the most preferred available relay, by probing multiple relays concurrently.

The parallel discovery logic of a Happy Eyeballs algorithm serves to reduce join latency for the initial join of an SSM channel. This

section and the preference ordering of relays defined in Section 2.3.2 taken together provide guidance on use of a Happy Eyeballs algorithm for the case of establishing AMT connections.

Note that according to the definition in Section 2.4.3 of this document, establishing the connection occurs before sending a membership report. As described in Section 5 of [RFC8085], only one of the successful connections will be used, and the others are all canceled or ignored. In the context of an AMT connection, this means the gateway will send the membership reports that subscribe to traffic only for the chosen connection, after the Happy Eyeballs algorithm resolves.

2.4.2. Algorithm Guidelines

During the "Initiation of asynchronous DNS queries" phase described in Section 3 of [RFC8305]), a gateway attempts to resolve the domain names listed in Section 2.3. This consists of resolving the SRV queries for DNS-SD domains for the AMT service, as well as the AMTRELAY query for the reverse IP domain defined in this document.

Each of the SRV and AMTRELAY responses might contain one or more IP addresses, (as with type 1 or type 2 AMTRELAY responses, or when the SRV Additional Data section of the SRV response contains the address records for the target, as urged by [RFC2782]), or they might contain only domain names (as with type 3 responses from Section 4.2.3, or an SRV response without an additional data section).

When present, IP addresses in the initial response provide resolved destination address candidates for the "Sorting of resolved destination addresses" phase described in Section 4 of [RFC8085]), whereas domain names without IP addresses in the initial response result in another set of queries for AAAA and A records, whose responses provide the candidate resolved destination addresses.

Since the SRV or AMTRELAY responses don't have a bound on the count of queries that might be generated aside from the bounds imposed by the DNS resolver, it's important for the gateway to provide a rate limit on the DNS queries. The DNS query functionality is expected to follow ordinary standards and best practices for DNS clients. A gateway MAY use an existing DNS client implementation that does so, and MAY rely on that client's rate limiting logic to avoid issuing excessive queries. Otherwise, a gateway MUST provide a rate limit for the DNS queries, and its default settings MUST NOT permit more than 10 queries for any 100-millisecond period (though this MAY be overridable by administrative configuration).

As the resolved IP addresses arrive, the Happy Eyeballs algorithm sorts them according to the requirements and recommendations given in Section 2.3.2, and attempts connections with the corresponding relays under the algorithm restrictions and guidelines given in [RFC8085] for the "Establishment of one connection, which cancels all other attempts" phase.

2.4.3. Connection Definition

Section 5 of [RFC8305] non-normatively describes success at a connection attempt as "generally when the TCP handshake completes".

There is no normative definition of a connection in the AMT specification [RFC7450], and there is no TCP connection involved in an AMT tunnel.

However, the concept of an AMT connection in the context of a Happy Eyeballs algorithm is a useful one, and so this section provides the following normative definition:

- o An AMT connection is completed successfully when the gateway receives from a newly discovered relay a valid Membership Query message (Section 5.1.4 of [RFC7450]) that does not have the L flag set.

See Section 2.5.5 of this document for further information about the relevance of the L flag to the establishment of a Happy Eyeballs connection. See Section 2.5.4 for an overview of how to respond if the connection does not provide multicast connectivity to the source.

To "cancel" this kind of AMT connection for the Happy Eyeballs algorithm, a gateway that has not sent a membership report with a subscription would simply stop sending AMT packets for that connection. A gateway only sends a membership report to a connection it has chosen as the most preferred available connection.

2.5. Guidelines for Restarting Discovery

2.5.1. Overview

It's expected that gateways deployed in different environments will use a variety of heuristics to decide when it's appropriate to restart the relay discovery process, in order to meet different performance goals (for example, to fulfill different kinds of service level agreements).

In general, restarting the discovery process is always safe for the gateway and relay during any of the events listed in this section,

but may cause a disruption in the forwarded traffic if the discovery process results in choosing a different relay, because this changes the RPF forwarding tree for the multicast traffic upstream of the gateway. This is likely to result in some dropped or duplicated packets from channels actively being tunneled from the old relay to the gateway.

The degree of impact on the traffic from choosing a different relay may depend on network conditions between the gateway and the new relay, as well as the network conditions and topology between the sender and the new relay, as this may cause the relay to propagate a new RPF join toward the sender.

Balancing the expected impact on the tunneled traffic against likely or observed problems with an existing connection to the relay is the goal of the heuristics that gateways use to determine when to restart the discovery process.

The non-normative advice in this section should be treated as guidelines to operators and implementors working with AMT systems that can use DRIAD as part of the relay discovery process.

2.5.2. Updates to Restarting Events

Section 5.2.3.4.1 of [RFC7450] lists several events that may cause a gateway to start or restart the discovery procedure.

This document provides some updates and recommendations regarding the handling of these and similar events. The first 5 events are copied here and numbered for easier reference, and the remaining 4 events are newly added for consideration in this document:

1. When a gateway pseudo-interface is started (enabled).
2. When the gateway wishes to report a group subscription when none currently exist.
3. Before sending the next Request message in a membership update cycle.
4. After the gateway fails to receive a response to a Request message.
5. After the gateway receives a Membership Query message with the L flag set to 1.

6. When the gateway wishes to report an (S,G) subscription with a source address that does not currently have other group subscriptions.
7. When there is a network change detected, for example when a gateway is operating inside an end user device or application, and the device joins a different network, or when the domain portion of a DNS-SD domain name changes in response to a DHCP message or administrative configuration.
8. When congestion or substantial loss is detected in the stream of AMT packets from a relay.
9. When the gateway has reported one or more (S,G) subscriptions, but no traffic is received from the source for some timeout. (See Section 2.5.4.1).

This list is not exhaustive, nor are any of the listed events strictly required to always force a restart of the discovery process.

Note that during event #1, a gateway may use DNS-SD, but does not have sufficient information to use DRIAD, since no source is known.

2.5.3. Tunnel Stability

In general, subscribers to active traffic flows that are being forwarded by an AMT gateway are less likely to experience a degradation in service (for example, from missing or duplicated packets) when the gateway continues using the same relay, as long as the relay is not overloaded and the network conditions remain stable.

Therefore, gateways SHOULD avoid performing a full restart of the discovery process during routine cases of event #3 (sending a new Request message), since it occurs frequently in normal operation.

However, see Section 2.5.4, Section 2.5.6, and Section 2.5.4.3 for more information about exceptional cases when it may be appropriate to use event #3.

2.5.4. Traffic Health

2.5.4.1. Absence of Traffic

If a gateway indicates one or more (S,G) subscriptions in a Membership Update message, but no traffic for any of the (S,G)s is received in a reasonable time, it's appropriate for the gateway to restart the discovery process.

If the gateway restarts the discovery process multiple times consecutively for this reason, the timeout period SHOULD be adjusted to provide a random exponential back-off.

The RECOMMENDED timeout is a random value in the range $[\text{initial_timeout}, \text{MIN}(\text{initial_timeout} * 2^{\text{retry_count}}, \text{maximum_timeout})]$, with a RECOMMENDED initial_timeout of 4 seconds and a RECOMMENDED maximum_timeout of 120 seconds.

Note that the recommended initial_timeout is larger than the initial timeout recommended in the similar algorithm from Section 5.2.3.4.3 of [RFC7450]. This is to provide time for RPF Join propagation in the sending network. Although the timeout values may be administratively adjusted to support performance requirements, operators are advised to consider the possibility of join propagation delays between the sender and the relay when choosing an appropriate timeout value.

Gateways restarting the discovery process because of an absence of traffic MUST use a hold-down timer that removes this relay from consideration during subsequent rounds of discovery while active. The hold-down SHOULD last for no less than 3 minutes and no more than 10 minutes.

2.5.4.2. Loss and Congestion

In some gateway deployments, it is also feasible to monitor the health of traffic flows through the gateway, for example by detecting the rate of packet loss by communicating out of band with receivers, or monitoring the packets of known protocols with sequence numbers. Where feasible, it's encouraged for gateways to use such traffic health information to trigger a restart of the discovery process during event #3 (before sending a new Request message).

However, to avoid synchronized rediscovery by many gateways simultaneously after a transient network event upstream of a relay results in many receivers detecting poor flow health at the same time, it's recommended to add a random delay before restarting the discovery process in this case.

The span of the random portion of the delay should be no less than 10 seconds by default, but may be administratively configured to support different performance requirements.

2.5.4.3. Ancient Discovery Information

In most cases, a gateway actively receiving healthy traffic from a relay that has not indicated load with the L flag should prefer to remain connected to the same relay, as described in Section 2.5.3.

However, a relay that appears healthy but has been forwarding traffic for days or weeks may have an increased chance of becoming unstable. Gateways may benefit from restarting the discovery process during event #3 (before sending a Request message) after the expiration of a long-term timeout, on the order of multiple hours, or even days in some deployments.

It may be beneficial for such timers to consider the amount of traffic currently being forwarded, and to give a higher probability of restarting discovery during periods with an unusually low data rate, to reduce the impact on active traffic while still avoiding relying on the results of a very old discovery.

Other issues may also be worth considering as part of this heuristic; for example, if the DNS expiry time of the record that was used to discover the current relay has not passed, the long term timer might be restarted without restarting the discovery process.

2.5.5. Relay Loaded or Shutting Down

The L flag (see Section 5.1.4.4 of [RFC7450]) is the preferred mechanism for a relay to signal overloading or a graceful shutdown to gateways.

A gateway that supports handling of the L flag should generally restart the discovery process when it processes a Membership Query packet with the L flag set. If an L flag is received while a concurrent Happy Eyeballs discovery process is under way for multiple candidate relays (Section 2.4), the relay sending the L flag SHOULD NOT be considered for the relay selection.

It is also RECOMMENDED that gateways avoid choosing a relay that has recently sent an L flag, with approximately a 10-minute hold-down. Gateways SHOULD treat this hold-down timer in the same way as the hold-down in Section 2.5.4.1, so that the relay is removed from consideration for short-term subsequent rounds of discovery.

2.5.6. Relay Discovery Messages vs. Restarting Discovery

All AMT relays are required by [RFC7450] to support handling of Relay Discovery messages (e.g. in Section 5.3.3.2 of [RFC7450]).

So a gateway with an existing connection to a relay can send a Relay Discovery message to the unicast address of that AMT relay. Under stable conditions with an unloaded relay, it's expected that the relay will return its own unicast address in the Relay Advertisement, in response to such a Relay Discovery message. Since this will not result in the gateway changing to another relay unless the relay

directs the gateway away, this is a reasonable exception to the advice against handling event #3 described in Section 2.5.3.

This behavior is discouraged for gateways that do support the L flag, to avoid sending unnecessary packets over the network.

However, gateways that do not support the L flag may be able to avoid a disruption in the forwarded traffic by sending such Relay Discovery messages regularly. When a relay is under load or has started a graceful shutdown, it may respond with a different relay address, which the gateway can use to connect to a different relay. This kind of coordinated handoff will likely result in a smaller disruption to the traffic than if the relay simply stops responding to Request messages, and stops forwarding traffic.

This style of Relay Discovery message (one sent to the unicast address of a relay that's already forwarding traffic to this gateway) SHOULD NOT be considered a full restart of the relay discovery process. It is RECOMMENDED for gateways to support the L flag, but for gateways that do not support the L flag, sending this message during event #3 may help mitigate service degradation when relays become unstable.

2.5.7. Independent Discovery Per Traffic Source

Relays discovered via the AMTRELAY RR are source-specific relay addresses, and may use different pseudo-interfaces from each other and from relays discovered via DNS-SD or a non-source-specific address, as described in Section 4.1.2.1 of [RFC7450].

Restarting the discovery process for one pseudo-interface does not require restarting the discovery process for other pseudo-interfaces. Gateway heuristics about restarting the discovery process should operate independently for different tunnels to relays, when responding to events that are specific to the different tunnels.

2.6. DNS Configuration

Often an AMT gateway will only have access to the source and group IP addresses of the desired traffic, and will not know any other name for the source of the traffic. Because of this, typically the best way of looking up AMTRELAY RRs will be by using the source IP address as an index into one of the reverse mapping trees (in-addr.arpa for IPv4, as described in Section 3.5 of [RFC1035], or ip6.arpa for IPv6, as described in Section 2.5 of [RFC3596]).

Therefore, it is RECOMMENDED that AMTRELAY RRs be added to reverse IP zones as appropriate. AMTRELAY records MAY also appear in other

zones, but the primary intended use case requires a reverse IP mapping for the source from an (S,G) in order to be useful to most AMT gateways.

When performing the AMTRELAY RR lookup, any CNAMEs or DNAMEs found MUST be followed. This is necessary to support zone delegation. Some examples outlining this need are described in [RFC2317].

See Section 4 and Section 4.3 for a detailed explanation of the contents for a DNS Zone file.

2.7. Waiting for DNS resolution

The DNS query functionality is expected to follow ordinary standards and best practices for DNS clients. A gateway MAY use an existing DNS client implementation that does so, and MAY rely on that client's retry logic to determine the timeouts between retries.

Otherwise, a gateway MAY re-send a DNS query if it does not receive an appropriate DNS response within some timeout period. If the gateway retries multiple times, the timeout period SHOULD be adjusted to provide a random exponential back-off.

As with the waiting process for the Relay Advertisement message from Section 5.2.3.4.3 of [RFC7450], the RECOMMENDED timeout is a random value in the range [initial_timeout, MIN(initial_timeout * 2^retry_count, maximum_timeout)], with a RECOMMENDED initial_timeout of 1 second and a RECOMMENDED maximum_timeout of 120 seconds.

3. Example Deployments

3.1. Example Receiving Networks

3.1.1. Tier 3 ISP

One example of a receiving network is an ISP that offers multicast ingest services to its subscribers, illustrated in Figure 3.

In the example network below, subscribers can join (S,G)s with MLDv2 or IGMPv3 as described in [RFC4604], and the AMT gateway in this ISP can receive and forward multicast traffic from one of the example sending networks in Section 3.2 by discovering the appropriate AMT relays with a DNS lookup for the AMTRELAY RR with the reverse IP of the source in the (S,G).

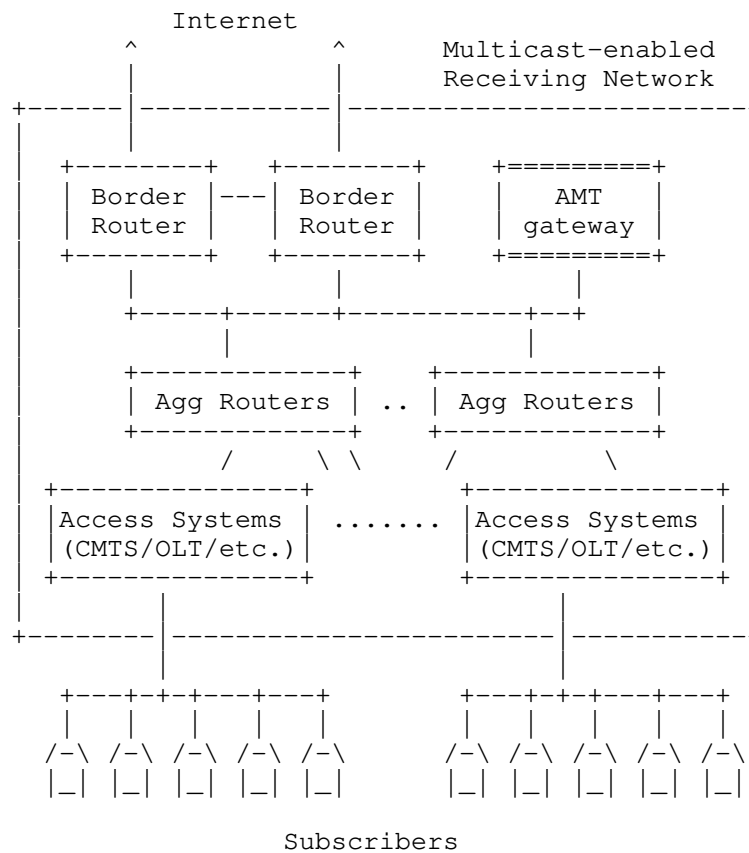


Figure 3: Receiving ISP Example

3.1.2. Small Office

Another example receiving network is a small branch office that regularly accesses some multicast content, illustrated in Figure 4.

This office has desktop devices that need to receive some multicast traffic, so an AMT gateway runs on a LAN with these devices, to pull traffic in through a non-multicast next-hop.

The office also hosts some mobile devices that have AMT gateway instances embedded inside apps, in order to receive multicast traffic over their non-multicast wireless LAN. (Note that the "Legacy Router" is a simplification that's meant to describe a variety of possible conditions; for example it could be a device providing a split-tunnel VPN as described in [RFC7359], deliberately excluding

multicast traffic for a VPN tunnel, rather than a device which is incapable of multicast forwarding.)

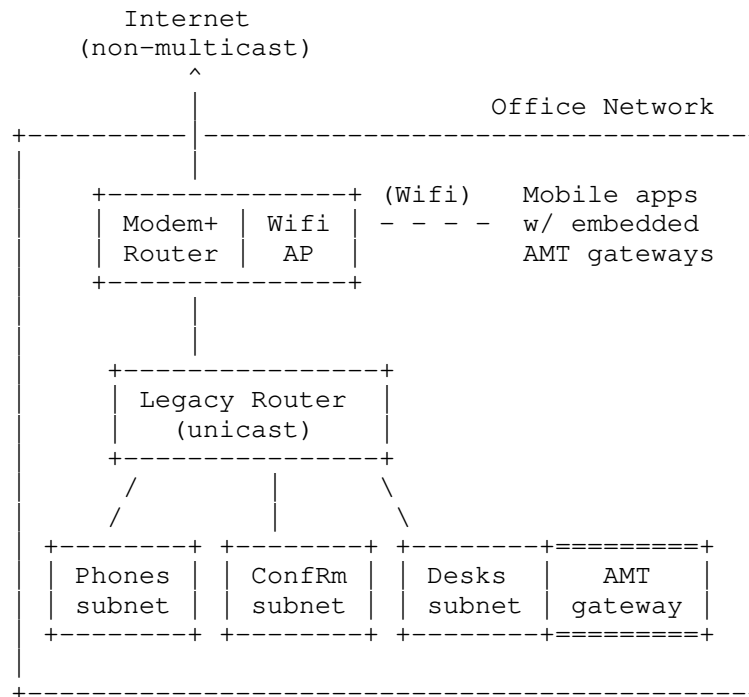


Figure 4: Small Office (no multicast up)

By adding an AMT relay to this office network as in Figure 5, it's possible to make use of multicast services from the example multicast-capable ISP in Section 3.1.1.

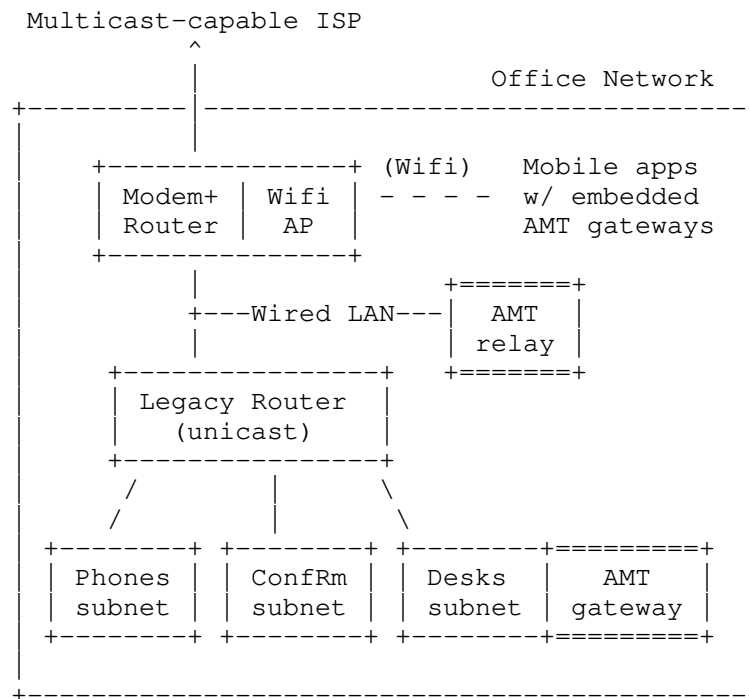


Figure 5: Small Office Example

When multicast-capable networks are chained like this, with a network like the one in Figure 5 receiving internet services from a multicast-capable network like the one in Figure 3, it's important for AMT gateways to reach the more local AMT relay, in order to avoid accidentally tunneling multicast traffic from a more distant AMT relay with unicast, and failing to utilize the multicast transport capabilities of the network in Figure 3.

3.2. Example Sending Networks

3.2.1. Sender-controlled Relays

When a sender network is also operating AMT relays to distribute multicast traffic, as in Figure 6, each address could appear as an AMTRELAY RR for the reverse IP of the sender, or one or more domain names could appear in AMTRELAY RRs, and the AMT relay addresses can be discovered by finding A or AAAA records from those domain names.

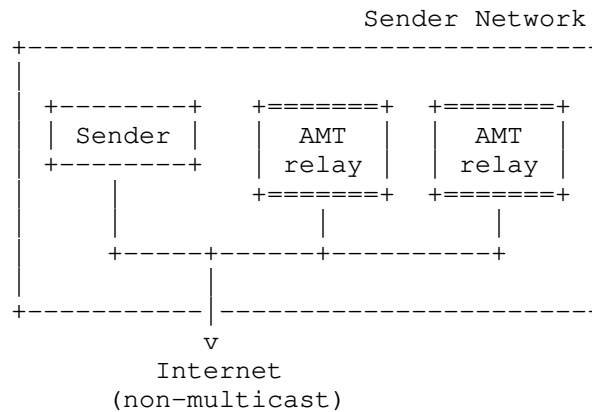


Figure 6: Small Office Example

3.2.2. Provider-controlled Relays

When an ISP offers a service to transmit outbound multicast traffic through a forwarding network, it might also offer AMT relays in order to reach receivers without multicast connectivity to the forwarding network, as in Figure 7. In this case it's RECOMMENDED that the ISP also provide at least one domain name for the AMT relays for use with the AMTRELAY RR.

When the sender wishes to use the relays provided by the ISP for forwarding multicast traffic, an AMTRELAY RR should be configured to use the domain name provided by the ISP, to allow for address reassignment of the relays without forcing the sender to reconfigure the corresponding AMTRELAY RRs.

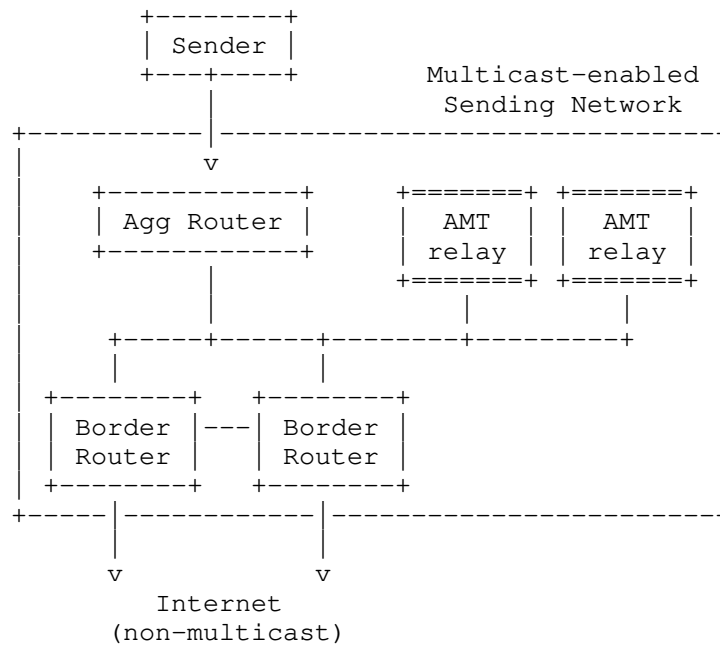


Figure 7: Sending ISP Example

4. AMTRELAY Resource Record Definition

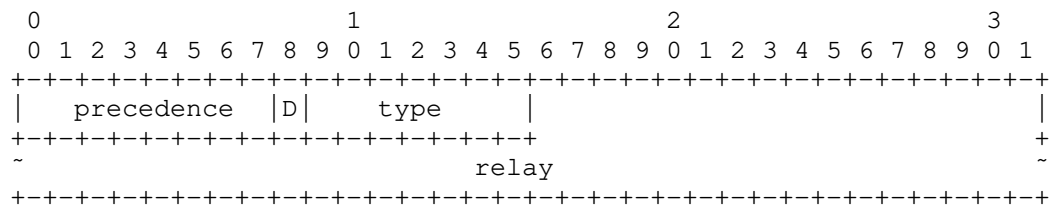
4.1. AMTRELAY RRTYPE

The AMTRELAY RRTYPE has the mnemonic AMTRELAY and type code 260 (decimal).

The AMTRELAY RR is class independent.

4.2. AMTRELAY RData Format

The AMTRELAY RData consists of a 8-bit precedence field, a 1-bit "Discovery Optional" field, a 7-bit type field, and a variable length relay field.



4.2.1. RData Format - Precedence

This is an 8-bit precedence for this record. It is interpreted in the same way as the PREFERENCE field described in Section 3.3.9 of [RFC1035].

Relays listed in AMTRELAY records with a lower value for precedence are to be attempted first.

4.2.2. RData Format - Discovery Optional (D-bit)

The D bit is a "Discovery Optional" flag.

If the D bit is set to 0, a gateway using this RR MUST perform AMT relay discovery as described in Section 4.2.1.1 of [RFC7450], rather than directly sending an AMT Request message to the relay.

That is, the gateway MUST receive an AMT Relay Advertisement message (Section 5.1.2 of [RFC7450]) for an address before sending an AMT Request message (Section 5.1.3 of [RFC7450]) to that address. Before receiving the Relay Advertisement message, this record has only indicated that the address can be used for AMT relay discovery, not for a Request message. This is necessary for devices that are not fully functional AMT relays, but rather load balancers or brokers, as mentioned in Section 4.2.1.1 of [RFC7450].

If the D bit is set to 1, the gateway MAY send an AMT Request message directly to the discovered relay address without first sending an AMT Discovery message.

This bit should be set according to advice from the AMT relay operator. The D bit MUST be set to zero when no information is available from the AMT relay operator about its suitability.

4.2.3. RData Format - Type

The type field indicates the format of the information that is stored in the relay field.

The following values are defined:

- o type = 0: The relay field is empty (0 bytes).
- o type = 1: The relay field contains a 4-octet IPv4 address.
- o type = 2: The relay field contains a 16-octet IPv6 address.

- o type = 3: The relay field contains a wire-encoded domain name. The wire-encoded format is self-describing, so the length is implicit. The domain name MUST NOT be compressed. (See Section 3.3 of [RFC1035] and Section 4 of [RFC3597].)

4.2.4. RData Format - Relay

The relay field is the address or domain name of the AMT relay. It is formatted according to the type field.

When the type field is 0, the length of the relay field is 0, and it indicates that no AMT relay should be used for multicast traffic from this source.

When the type field is 1, the length of the relay field is 4 octets, and a 32-bit IPv4 address is present. This is an IPv4 address as described in Section 3.4.1 of [RFC1035]. This is a 32-bit number in network byte order.

When the type field is 2, the length of the relay field is 16 octets, and a 128-bit IPv6 address is present. This is an IPv6 address as described in Section 2.2 of [RFC3596]. This is a 128-bit number in network byte order.

When the type field is 3, the relay field is a normal wire-encoded domain name, as described in Section 3.3 of [RFC1035]. Compression MUST NOT be used, for the reasons given in Section 4 of [RFC3597].

For a type 3 record, the D-bit and preference fields carry over to all A or AAAA records for the domain name. There is no difference in the result of the discovery process when it's obtained by type 1 or type 2 AMTRELAY records with identical D-bit and preference fields, vs. when the result is obtained by a type 3 AMTRELAY record that resolves to the same set of IPv4 and IPv6 addresses via A and AAAA lookups.

4.3. AMTRELAY Record Presentation Format

4.3.1. Representation of AMTRELAY RRs

AMTRELAY RRs may appear in a zone data master file. The precedence, D-bit, relay type, and relay fields are REQUIRED.

If the relay type field is 0, the relay field MUST be ".".

The presentation for the record is as follows:

IN AMTRELAY precedence D-bit type relay

4.3.2. Examples

In a DNS authoritative nameserver that understands the AMTRELAY type, the zone might contain a set of entries like this:

```
$ORIGIN 100.51.198.in-addr.arpa.
10      IN AMTRELAY 10 0 1 203.0.113.15
10      IN AMTRELAY 10 0 2 2001:DB8::15
10      IN AMTRELAY 128 1 3 amtrelays.example.com.
```

This configuration advertises an IPv4 discovery address, an IPv6 discovery address, and a domain name for AMT relays which can receive traffic from the source 198.51.100.10. The IPv4 and IPv6 addresses are configured with a D-bit of 0 (meaning discovery is mandatory, as described in Section 4.2.2), and a precedence 10 (meaning they're preferred ahead of the last entry, which has precedence 128).

For zone files in name servers that don't support the AMTRELAY RRTYPE natively, it's possible to use the format for unknown RR types, as described in [RFC3597]. This approach would replace the AMTRELAY entries in the example above with the entries below:

```
10      IN TYPE260 \# (
        6 ; length
        0a ; precedence=10
        01 ; D=0, relay type=1, an IPv4 address
        cb00710f ) ; 203.0.113.15
10      IN TYPE260 \# (
        18 ; length
        0a ; precedence=10
        02 ; D=0, relay type=2, an IPv6 address
        20010db8000000000000000000000000f ) ; 2001:db8::15
10      IN TYPE260 \# (
        24 ; length
        80 ; precedence=128
        83 ; D=1, relay type=3, a wire-encoded domain name
        09616d74726556c617973076578616d706c6503636f6d ) ; domain name
```

See Appendix A for more details.

5. IANA Considerations

This document updates the IANA Registry for DNS Resource Record Types by assigning type 260 to the AMTRELAY record.

This document creates a new registry named "AMTRELAY Resource Record Parameters", with a sub-registry for the "Relay Type Field". The initial values in the sub-registry are:

Value	Description
0	No relay is present.
1	A 4-byte IPv4 address is present
2	A 16-byte IPv6 address is present
3	A wire-encoded domain name is present
4-255	Unassigned

Values 0, 1, 2, and 3 are further explained in Section 4.2.3 and Section 4.2.4. Relay type numbers 4 through 255 can be assigned with a policy of Specification Required (as described in [RFC8126]).

6. Security Considerations

6.1. Use of AMT

This document defines a mechanism that enables a more widespread and automated use of AMT, even without access to a multicast backbone. Operators of networks and applications that include a DRIAD-capable AMT gateway are advised to carefully consider the security considerations in Section 6 of [RFC7450].

AMT gateway operators also are encouraged to take appropriate steps to ensure the integrity of the data received via AMT, for example by the opportunistic use of IPSec [RFC4301] to secure traffic received from AMT relays, when IPSECKEY records [RFC4025] are available or when a trust relationship with the AMT relays can be otherwise established and secured.

6.2. Record-spoofing

The AMTRELAY resource record contains information that SHOULD be communicated to the DNS client without being modified. The method used to ensure the result was unmodified is up to the client.

There must be a trust relationship between the end consumer of this resource record and the DNS server. This relationship may be end-to-end DNSSEC validation, a TSIG [RFC2845] or SIG(0) [RFC2931] channel to another secure source, a secure local channel on the host, DNS over TLS [RFC7858] or HTTPS [RFC8484], or some other secure mechanism.

If an AMT gateway accepts a maliciously crafted AMTRELAY record, the result could be a Denial of Service, or receivers processing multicast traffic from a source under the attacker's control.

6.3. Congestion

Multicast traffic, particularly interdomain multicast traffic, carries some congestion risks, as described in Section 4 of [RFC8085].

Application implementors and network operators that use DRIAD-capable AMT gateways are advised to take precautions including monitoring of application traffic behavior, traffic authentication at ingest, rate-limiting of multicast traffic, and the use of circuit-breaker techniques such as those described in Section 3.1.10 of [RFC8085] and similar protections at the network level, in order to ensure network health in the event of misconfiguration, poorly written applications that don't follow UDP congestion control principles, or deliberate attack.

7. Acknowledgements

This specification was inspired by the previous work of Doug Nortz, Robert Sayko, David Segelstein, and Percy Tarapore, presented in the MBONED working group at IETF 93.

Thanks to Jeff Goldsmith, Toerless Eckert, Mikael Abrahamsson, Lenny Giuliano, Mark Andrews, Sandy Zheng, Kyle Rose, Ben Kaduk, Bill Atwood, Tim Chown, and Warren Kumari for their very helpful comments.

8. References

8.1. Normative References

- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <<https://www.rfc-editor.org/info/rfc1034>>.
- [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>>.
- [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>>.
- [RFC2181] Elz, R. and R. Bush, "Clarifications to the DNS Specification", RFC 2181, DOI 10.17487/RFC2181, July 1997, <<https://www.rfc-editor.org/info/rfc2181>>.

- [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>>.
- [RFC3376] Cain, B., Deering, S., Kouvelas, I., Fenner, B., and A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, DOI 10.17487/RFC3376, October 2002, <<https://www.rfc-editor.org/info/rfc3376>>.
- [RFC3596] Thomson, S., Huitema, C., Ksinant, V., and M. Souissi, "DNS Extensions to Support IP Version 6", STD 88, RFC 3596, DOI 10.17487/RFC3596, October 2003, <<https://www.rfc-editor.org/info/rfc3596>>.
- [RFC3597] Gustafsson, A., "Handling of Unknown DNS Resource Record (RR) Types", RFC 3597, DOI 10.17487/RFC3597, September 2003, <<https://www.rfc-editor.org/info/rfc3597>>.
- [RFC3810] Vida, R., Ed. and L. Costa, Ed., "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, DOI 10.17487/RFC3810, June 2004, <<https://www.rfc-editor.org/info/rfc3810>>.
- [RFC4604] Holbrook, H., Cain, B., and B. Haberman, "Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast", RFC 4604, DOI 10.17487/RFC4604, August 2006, <<https://www.rfc-editor.org/info/rfc4604>>.
- [RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, DOI 10.17487/RFC4607, August 2006, <<https://www.rfc-editor.org/info/rfc4607>>.
- [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, <<https://www.rfc-editor.org/info/rfc6724>>.
- [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>>.
- [RFC7450] Bumgardner, G., "Automatic Multicast Tunneling", RFC 7450, DOI 10.17487/RFC7450, February 2015, <<https://www.rfc-editor.org/info/rfc7450>>.

- [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>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8305] Schinazi, D. and T. Pauly, "Happy Eyeballs Version 2: Better Connectivity Using Concurrency", RFC 8305, DOI 10.17487/RFC8305, December 2017, <<https://www.rfc-editor.org/info/rfc8305>>.

8.2. Informative References

- [RFC2317] Eidnes, H., de Groot, G., and P. Vixie, "Classless IN-ADDR.ARPA delegation", BCP 20, RFC 2317, DOI 10.17487/RFC2317, March 1998, <<https://www.rfc-editor.org/info/rfc2317>>.
- [RFC2845] Vixie, P., Gudmundsson, O., Eastlake 3rd, D., and B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)", RFC 2845, DOI 10.17487/RFC2845, May 2000, <<https://www.rfc-editor.org/info/rfc2845>>.
- [RFC2931] Eastlake 3rd, D., "DNS Request and Transaction Signatures (SIG(0)s)", RFC 2931, DOI 10.17487/RFC2931, September 2000, <<https://www.rfc-editor.org/info/rfc2931>>.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <<https://www.rfc-editor.org/info/rfc3550>>.
- [RFC4025] Richardson, M., "A Method for Storing IPsec Keying Material in DNS", RFC 4025, DOI 10.17487/RFC4025, March 2005, <<https://www.rfc-editor.org/info/rfc4025>>.
- [RFC4301] Kent, S. and K. Seo, "Security Architecture for the Internet Protocol", RFC 4301, DOI 10.17487/RFC4301, December 2005, <<https://www.rfc-editor.org/info/rfc4301>>.
- [RFC5110] Savola, P., "Overview of the Internet Multicast Routing Architecture", RFC 5110, DOI 10.17487/RFC5110, January 2008, <<https://www.rfc-editor.org/info/rfc5110>>.

- [RFC6726] Paila, T., Walsh, R., Luby, M., Roca, V., and R. Lehtonen, "FLUTE - File Delivery over Unidirectional Transport", RFC 6726, DOI 10.17487/RFC6726, November 2012, <<https://www.rfc-editor.org/info/rfc6726>>.
- [RFC7359] Gont, F., "Layer 3 Virtual Private Network (VPN) Tunnel Traffic Leakages in Dual-Stack Hosts/Networks", RFC 7359, DOI 10.17487/RFC7359, August 2014, <<https://www.rfc-editor.org/info/rfc7359>>.
- [RFC7761] Fenner, B., Handley, M., Holbrook, H., Kouvelas, I., Parekh, R., Zhang, Z., and L. Zheng, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", STD 83, RFC 7761, DOI 10.17487/RFC7761, March 2016, <<https://www.rfc-editor.org/info/rfc7761>>.
- [RFC7858] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)", RFC 7858, DOI 10.17487/RFC7858, May 2016, <<https://www.rfc-editor.org/info/rfc7858>>.
- [RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [RFC8313] Tarapore, P., Ed., Sayko, R., Shepherd, G., Eckert, T., Ed., and R. Krishnan, "Use of Multicast across Inter-domain Peering Points", BCP 213, RFC 8313, DOI 10.17487/RFC8313, January 2018, <<https://www.rfc-editor.org/info/rfc8313>>.
- [RFC8484] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <<https://www.rfc-editor.org/info/rfc8484>>.
- [RFC8499] Hoffman, P., Sullivan, A., and K. Fujiwara, "DNS Terminology", BCP 219, RFC 8499, DOI 10.17487/RFC8499, January 2019, <<https://www.rfc-editor.org/info/rfc8499>>.

Appendix A. Unknown RRTYPE construction

In a DNS resolver that understands the AMTRELAY type, the zone file might contain this line:

```
IN AMTRELAY 128 0 3 amtrelays.example.com.
```

In order to translate this example to appear as an unknown RRTYPE as defined in [RFC3597], one could run the following program:

```
<CODE BEGINS>
$ cat translate.py
#!/usr/bin/env python3
import sys
name=sys.argv[1]
wire=''
for dn in name.split('.'):
    if len(dn) > 0:
        wire += ('%02x' % len(dn))
        wire += (''.join('%02x'%ord(x) for x in dn))
print(len(wire)//2) + 2
print(wire)

$ ./translate.py amtrelays.example.com
24
09616d74726556c617973076578616d706c6503636f6d
<CODE ENDS>
```

The length and the hex string for the domain name "amtrelays.example.com" are the outputs of this program, yielding a length of 22 and the above hex string.

22 is the length of the wire-encoded domain name, so to this we add 2 (1 for the precedence field and 1 for the combined D-bit and relay type fields) to get the full length of the RData, and encode the precedence, D-bit, and relay type fields as octets, as described in Section 4.

This results in a zone file entry like this:

```
IN TYPE260 \# ( 24 ; length
      80 ; precedence = 128
      03 ; D-bit=0, relay type=3 (wire-encoded domain name)
      09616d74726556c617973076578616d706c6503636f6d ) ; domain name
```

Author's Address

Jake Holland
Akamai Technologies, Inc.
150 Broadway
Cambridge, MA 02144
United States of America

Email: jakeholland.net@gmail.com

Mboned
Internet-Draft
Intended status: Standards Track
Expires: March 29, 2020

J. Holland
K. Rose
Akamai Technologies, Inc.
September 26, 2019

Asymmetric Manifest Based Integrity
draft-jholland-mboned-ambi-04

Abstract

This document defines Asymmetric Manifest-Based Integrity (AMBI). AMBI allows each receiver or forwarder of a stream of multicast packets to check the integrity of the contents of each packet in the data stream. AMBI operates by passing cryptographically verifiable hashes of the data packets inside manifest messages, and sending the manifests over authenticated out-of-band communication channels.

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 <https://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 March 29, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

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

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

Table of Contents

1. Introduction	2
1.1. Comparison with TESLA	4
1.2. Terminology	5
2. Protocol Operation	5
2.1. Overview	5
2.2. Buffering and Validation Windows	5
2.2.1. Inter-packet Gap	7
2.3. Packet Digests	7
2.3.1. Digest Profile	7
2.3.2. Pseudoheader	10
2.4. Manifests	12
2.4.1. Manifest Layout	12
2.5. Transitioning to Other Manifest Streams	14
3. Transport Considerations	14
3.1. Overview	14
3.2. HTTPS	15
3.3. DTLS	15
3.4. DTLS + FECFRAME	15
4. Examples	15
5. YANG Module	15
5.1. Tree Diagram	16
5.2. Module	16
6. IANA Considerations	18
6.1. The YANG Module Names Registry	18
6.2. Media Type	19
7. Security Considerations	19
7.1. Predictable Packets	19
8. Acknowledgements	19
9. References	19
9.1. Normative References	19
9.2. Informative References	20
Authors' Addresses	21

1. Introduction

Multicast transport poses security problems that are not easily addressed by the same security mechanisms used for unicast transport.

The "Introduction" sections of the documents describing TESLA [RFC4082], and TESLA in SRTP [RFC4383], and TESLA with ALC and NORM [RFC5776] present excellent overviews of the challenges unique to multicast authentication, briefly summarized here:

- o A MAC based on a symmetric shared secret cannot be used because each packet has multiple receivers that do not trust each other, and using a symmetric shared secret exposes the same secret to each receiver.
- o Asymmetric per-packet signatures can handle only very low bit-rates because of the computational overhead.
- o An asymmetric signature of a larger message comprising multiple packets requires reliable receipt of all such packets, something that cannot be guaranteed in a timely manner even for protocols that do provide reliable delivery, and the retransmission of which may anyway exceed the useful lifetime for data formats that can otherwise tolerate some degree of loss.

Asymmetric Manifest-Based Integrity (AMBI) defines a method for receivers or middle boxes to cryptographically authenticate and verify the integrity of a stream of packets, by communicating packet "manifests" (described in Section 2.4) via an out-of-band communication channel that provides authentication and verifiable integrity.

Each manifest contains a message digest (described in Section 2.3) for each packet in a sequence of packets from the data stream, hereafter called a "packet digest". The packet digest incorporates a cryptographic hash of the packet contents and some identifying data from the packet, according to a defined digest profile for the data stream.

Each manifest MUST be delivered in a way that provides cryptographic integrity guarantees of the authenticity of the manifest. For example, TLS could be used to deliver a stream of manifests over a unicast data stream from a set of trusted senders to each receiver, or a protocol that asymmetrically signs each message could be used to transport authenticated manifests over a multicast channel. Note that a UDP-based protocol might drop or reorder manifests while still providing authentication.

Upon successful verification of a manifest and receipt of any subset of the corresponding data packets, the receiver has proof of the integrity of the contents of the data packets that are listed in the manifest.

Authenticating the integrity of the data packets depends on:

- o the authenticity of the manifests; and

- o the authenticity of the digest profile used for construction of the packet digests; and
- o the difficulty of generating a collision for the packet digests contained in the manifest.

This document defines a YANG [RFC7950] module that augments the DORMS [I-D.draft-jholland-mboned-dorms-00] YANG module to provide a way to communicate a digest profile, described in Section 2.3.1, for construction of the packet digests, described in Section 2.3. When obtaining the digest profile by using DORMS, the authenticity of the data stream relies on a trust relationship with the DORMS server, since that anchors the authenticity of the digest profile for constructing packet digests.

1.1. Comparison with TESLA

AMBI and TESLA [RFC4082] and [RFC5776] attempt to achieve a similar goal of authenticating the integrity of streams of multicast packets. AMBI imposes a higher overhead, as measured in the amount of extra data required, than TESLA imposes. In exchange, AMBI provides non-repudiation (which TESLA does not), and relaxes the requirement for establishing an upper bound on clock synchronization between sender and receiver.

This tradeoff enables new capabilities for AMBI, relative to TESLA. In particular, when receiving multicast traffic from an untrusted transit network, AMBI can be used by a middle box to authenticate packets from a trusted source before forwarding traffic through the network, and the receiver also can separately authenticate the packets it receives.

This use case is not possible with TESLA because the data packets can't be authenticated until a key is disclosed, so either the middlebox has to forward data packets without first authenticating so that the receiver has them prior to key disclosure, or the middlebox has to hold packets until the key is disclosed, at which point the receiver can no longer establish their authenticity.

The other new capability is that because AMBI provides authentication information out of band, authentication can be retrofitted into some pre-existing deployments without changing the protocol of the data packets, under some restrictions outlined in Section 7. By contrast, TESLA requires a MAC to be added to each authenticated message.

1.2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] and [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Protocol Operation

2.1. Overview

In order to authenticate a data packet, AMBI receivers need to hold these three pieces of information at the same time:

- o the data packet; and
- o an authenticated manifest containing the packet digest for the data packet; and
- o a digest profile defining the transformation from the data packet to its packet digest.

The manifests are delivered as a stream of manifests over an authenticated data channel. Manifest contents **MUST** be authenticated before they can be used to authenticate data packets.

The manifest stream is composed of an ordered sequence of manifests that each contain an ordered sequence of packet digests, corresponding to the original packets as sent from their origin, in the same order.

2.2. Buffering and Validation Windows

Using different communication channels for the manifest stream and the data stream introduces a possibility of desynchronization in the timing of the received data between the different channels, so receivers hold data packets and packet digests from the manifest stream in buffers for some duration while awaiting the arrival of their counterparts.

While holding a data packet, if the corresponding packet digest for that packet arrives in the manifest stream and can be authenticated, the data packet is authenticated.

While holding an authenticated packet digest, if the corresponding data packet arrives with a matching packet digest, the data packet is authenticated.

Once a data packet is authenticated, the corresponding packet digest can be discarded and the data packet can be further processed by the receiving application or forwarded through the receiving network. Authenticating a data packet consumes one packet digest and prevents re-learning, with a hold-down time equal to the hold time for packet digests. A different manifest might provide the same packet digest with the same packet sequence number, but the digest remains consumed if it has been used to authenticate a data packet.

If the receiver's hold duration for a data packet expires without authenticating the packet, the packet SHOULD be dropped as unauthenticated. If the hold duration of a manifest expires, packet digests last received in that manifest SHOULD be discarded. (Note that in some cases, packet digests can be sent redundantly in more than one manifest. In such cases, the latest received time for an authenticated packet digest should be used for the expiration time.)

Since packet digests are usually smaller than the data packets, it's RECOMMENDED that senders generate and send manifests with timing such that the packet digests in a manifest will typically be received by subscribed receivers before the data packets corresponding to those digests are received.

This strategy reduces the buffering requirements at receivers at, the cost of introducing some buffering of data packets at the sender, since data packets are generated before their packet digests can be added to manifests.

The RECOMMENDED default hold times at receivers are:

- o 2 seconds for data packets
- o 10 seconds for packet digests

The sender MAY recommend different values for specific data streams, in order to tune different data streams for different performance goals. The YANG model in Section 5 provides a mechanism for senders to communicate the sender's recommendation for buffering durations, when using DORMS.

Receivers SHOULD follow the recommendations for hold times provided by the sender, subject to their capabilities and any administratively configured limits on buffer sizes at the receiver.

However receivers MAY deviate from the values recommended by the sender for a variety of reasons. Decreasing the buffering durations recommended by the server increases the risk of losing packets, but

can be an appropriate tradeoff for specific network conditions and hardware constraints on some devices.

TBD: should there be any reordering restrictions above and beyond the timing constraints?

2.2.1. Inter-packet Gap

It's RECOMMENDED that middle boxes forwarding buffered data packets preserve the inter-packet gap between packets, and that receiving libraries provide mechanisms to expose the network arrival times of packets to applications.

The purpose for this recommendation is to preserve the capability of receivers to use techniques for available bandwidth detection or network congestion based on observation of packet times. Examples of such techniques include paced chirping and pathrate.

Note that this recommendation SHOULD NOT prevent the transmission of an authenticated packet because the prior packet is unauthenticated. This recommendation only asks implementations to delay the transmission of an authenticated packet to correspond to the interpacket gap if an authenticated packet was previously transmitted and the authentication of the subsequent packet would otherwise burst the packets more quickly.

This does not prevent the transmission of packets out of order according to their order of authentication, only the timing of packets that are transmitted, after authentication, in the same order they were received.

For receiver applications, the time that the original packet was received from the network SHOULD be made available to the receiving application.

2.3. Packet Digests

2.3.1. Digest Profile

A packet digest is a message digest for a data packet, built according to a digest profile defined by the sender.

The digest profile is defined by the sender, and specifies:

1. A cryptographically secure hash algorithm (REQUIRED)
2. A manifest stream identifier

3. Whether to hash the IP payload or the UDP payload. (see Section 2.3.1.1)

The hash algorithm is applied to a pseudoheader followed by the packet payload, as determined by the digest profile. The computed hash value is the packet digest.

TBD: there should also be a way to specify that only packets to a specific UDP port are applicable. I think this is not quite right today and probably should be done with a grouping in the yang model, so that the profile appears either inside a "protocol" container inside the (S,G) or inside the udp-stream inside the "protocol", but am not sure. Follow-up on this after the first reference implementation...

2.3.1.1. Payload Type

2.3.1.1.1. UDP vs. IP payload validation

When the digest profile indicates that UDP payloads are validated, the IP protocol for the packets MUST be UDP (0x11) and the payload used for calculating the packet digest includes only the UDP payload, with length as the number of UDP payload octets, as calculated by subtracting the size of the UDP header from the UDP payload length.

When the digest profile indicates that IP payloads are validated, the IP payload of the packet is used, using the outermost IP layer that contains the (S,G) corresponding to the (S,G) protected by the manifest. There is no restriction on the IP protocols that can be authenticated. The length field in the pseudoheader is calculated by subtracting the IP Header Length from the IP length, and is equal to the number of octets in the payload for the digest calculation.

2.3.1.1.2. Motivation

Full IP payloads often aren't available to receivers without extra privileges on end user operating systems, so it's useful to provide a way to authenticate only the UDP payload, which is often the only portion of the packet available to many receiving applications.

However, for some use cases a full IP payload is appropriate. For example, when retrofitting some existing protocols, some packets may be predictable or frequently repeated. Use of an IPSec Authentication Header [RFC4302] is one way to disambiguate such packets. Even though the shared secret means the Authentication Header can't itself be used to authenticate the packet contents, the sequence number in the Authentication Header can ensure that specific

packets are not repeated at the IP layer, and so it's useful for AMBI to have the capability to authenticate such packets.

Another example: some services might need to authenticate the UDP options [I-D.ietf-tsvwg-udp-options]. When using the UDP payload, the UDP options would not be part of the authenticated payload, but would be included when using the IP payload type.

Lastly, since (S,G) subscription operates at the IP layer, it's possible that some non-UDP protocols will need to be authenticated.

2.3.1.2. TBD: Packet contents?

TBD: Determine whether we need to support packet contents in the packet digest. If so, add to above list in Section 2.3.1:

- o A set of bits from the packet contents (potentially empty)

The packet contents are a sequence of bits composed from a sequence of fixed bit (offset, length) pairs, as specified in xxxxxx. A useful choice for packet contents is to use sequence numbers in the application level protocol, such as with RTP [RFC3550], but any contents from the packet with a fixed bit offset and length can be used.

Providing variable packet contents in the packet digest increases the difficulty of attacking the hash by limiting the scope of legitimate data packets that can be matched when attempting to generate a hash collision.

The basic idea is to put an encoding here so that for example the RTP sequence number or the sequence number in an Authentication Header can be provided here in bulk (you give "value starts at bit 80 and is 16 bits long unsigned and increases by 1 per packet for the packets in the manifest with starting value 10", indicating that the 100 packets in the manifest have values 10-110 in their contents at the given location. Now those contents are prepended to the packet digest, and can be verified against the packets, as well as the hash of the contents).

For packet streams without a sequence number, we can instead incorporate a few high-entropy bits from the packet contents and NOT provide the value as a sequence number, but rather incorporate it in the digest values themselves. (Is this useful?)

Before defining this, I want to calculate how much overhead it buys us- how much can we truncate a good hash algorithm if we use this to

add collision resistance? Might not be worthwhile, it's a significant increase in complexity. -jake 2019-08-31

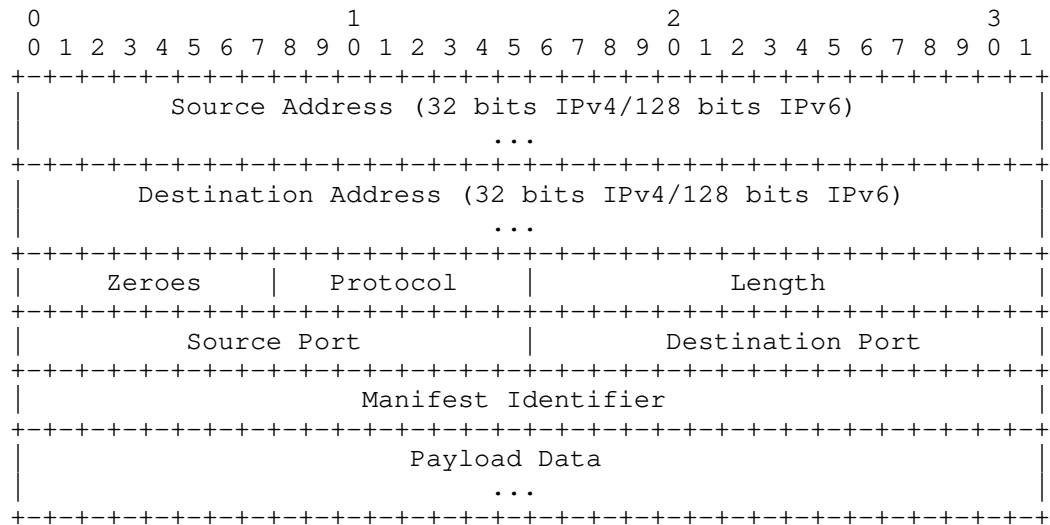
If we need it, tentative addition to yang for the data profile looks like:

```
list packet-contents {
  key offset;
  description "contents from the packet for the packet
    digest";
  leaf offset {
    type uint16;
    mandatory true;
    description "offset of the contents, in number of bits";
  }
  leaf length {
    type uint16;
    mandatory true;
    description "length of the contents, in number of bits";
  }
  leaf manifest-delivery {
    type enumeration {
      enum sequence;
      enum digest;
    }
    mandatory true;
    description "the way these content bits are delivered in
      the manifest";
  }
}
```

The manifest-delivery would indicate whether the bits are a sequence number (in which case a section for a manifest with a start+step would be added ahead of the digests), or digest (indicating the bits appear inside each digest, ahead of the hash), and they would prepend in order to the packet digest, with sequence number bits inserted at the right bit location for the digest, based on earlier-appearing values, if any.

2.3.2. Pseudoheader

When calculating the hash for the packet digest, the hash algorithm is applied to a pseudoheader followed by the payload from the packet. The complete sequence of octets used to calculate the hash is structured as follows:



2.3.2.1. Source Address

The IPv4 or IPv6 source address of the packet.

2.3.2.2. Destination Address

The IPv4 or IPv6 destination address of the packet.

2.3.2.3. Zeroes

All bits set to 0.

2.3.2.4. Protocol

The IP Protocol field from IPv4, or the Next Header field for IPv6. When UDP payload is indicated, this value MUST be UDP (0x11).

2.3.2.5. Length

The length in octets of the Payload Data field, expressed as an unsigned 16-bit integer.

2.3.2.6. Source Port

The source port of the packet. Zeroes if using a protocol that does not use source ports.

2.3.2.7. Destination Port

The destination port of the packet. Zeroes if using a protocol that does not use destination ports.

TBD: there's something I hate about the source and destination ports. Maybe it should only be active in UDP-payload mode, instead of zeroes when not UDP? But I suspect there's a better approach than UDP-or-not, so it's this way for now, with hopes of finding something better in the next version.

2.3.2.8. Manifest Identifier

The 32-bit identifier for the manifest stream.

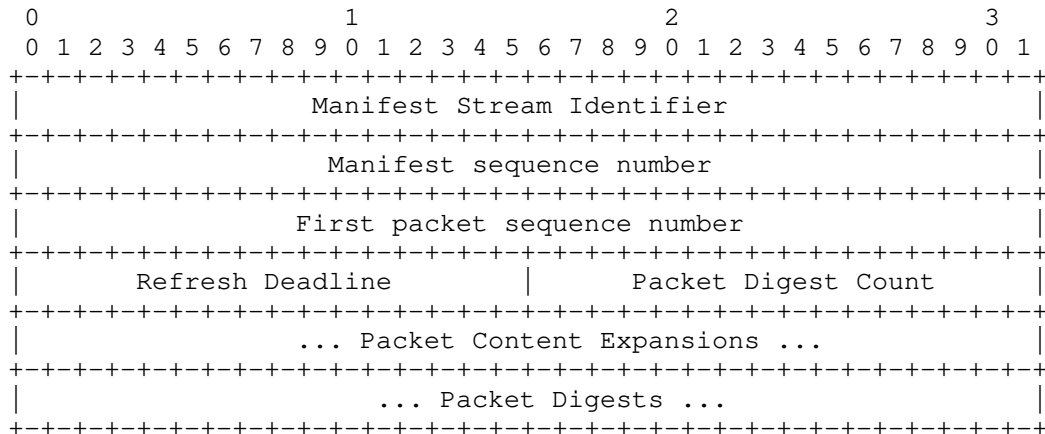
2.3.2.9. Payload Data

The payload data includes either the IP payload or the UDP payload, as indicated by the digest profile.

The payload type is configurable because when sending UDP, some legacy networks may strip the UDP option space, and it's necessary to provide a manifest stream capable of authentication that can interoperate with these networks. However, for non-UDP traffic or in order to authenticate the UDP options, some use cases may require support for authenticating the full IP payload.

2.4. Manifests

2.4.1. Manifest Layout



2.4.1.1. Manifest Stream Identifier

A 32-bit unsigned integer chosen by the sender.

2.4.1.2. Manifest Sequence Number

A monotonically increasing 32-bit unsigned integer. Each manifest sent by the sender increases this value by 1. On overflow it wraps to 0.

It's RECOMMENDED to expire the manifest stream and start a new stream for the data packets before a sequence number wrap is necessary.

2.4.1.3. First Packet Sequence Number

A monotonically increasing 32-bit unsigned integer. Each packet in the data stream increases this value by 1.

It's RECOMMENDED to expire the manifest stream and start a new stream for the data packets before a sequence number wrap is necessary.

Note: for redundancy, especially if using a manifest stream with unreliable transport, successive manifests MAY provide duplicates of the same packet digest with the same packet sequence number, using overlapping sets of packet sequence numbers. When received, these reset the hold timer for the listed packet digests.

2.4.1.4. Refresh Deadline

A 16-bit unsigned integer number of seconds.

A zero value means the current digest profile for the current manifest stream is stable.

A nonzero value means that the authentication is transitioning to a new manifest stream, and the set of digest profiles SHOULD be refreshed by receivers that might stay joined longer than this duration, and a different manifest stream SHOULD be selected, before this many seconds have elapsed, in order to avoid a disruption. See Section 2.5.

2.4.1.5. Packet Digest Count

The count of packet digests in the manifest.

2.4.1.6. Packet Digests

Packet digests appended one after the other, aligned to 8-bit boundaries with zero padding (if the bit length of the digests are not multiples of 8 bits).

2.5. Transitioning to Other Manifest Streams

It's possible for multiple manifest streams authenticating the same data stream to be active at the same time. The different manifest streams can have different hash algorithms, manifest ids, and current packet sequence numbers for the same data stream. These result in different sets of packet digests for the same data packets, one digest per packet per digest profile.

It's necessary sometimes to transition gracefully from one manifest stream to another. The Refresh Deadline field from the manifest is used to signal to receivers the need to transition.

When a receiver gets a nonzero refresh deadline in a manifest the sender SHOULD have an alternate manifest stream ready and available, and the receiver SHOULD learn the alternate manifest stream, join the new one, and leave the old one before the number of seconds given in the refresh deadline. After the refresh deadline has expired, a manifest stream MAY end.

The receivers SHOULD use a random value between now and one half the number of seconds in the deadline field, to spread the spike of load on the DORMS server during a large multicast event.

3. Transport Considerations

3.1. Overview

AMBI manifests MUST be authenticated, but any transport protocol providing authentication can be used. This section discusses several viable options for the use of an authenticating transport, and some associated design considerations.

TBD: extend the 'manifest-transport' in the YANG model to make an extensible mechanism to advertise different transport options for receiving manifest streams.

TBD: add ALTA to the list when and if it gets further along [I-D.draft-krose-mboned-alta-01]. Sending an authenticatable multicast stream (instead of the below unicast-based proposals) is a worthwhile goal, else a 1% unicast authentication overhead becomes a new unicast limit to the scalability.

3.2. HTTPS

This document defines a new media type 'application/ambi' for use with HTTPS.

An HTTPS stream carrying the 'application/ambi' media type is composed of a sequence of binary AMBI manifests. It is RECOMMENDED to use Chunked encoding.

Complete packet Digests from partially received manifests MAY be used by the receiver for authentication, even if the full manifest is not yet delivered.

3.3. DTLS

TBD: DTLS [RFC6347] can provide authentication for datagrams, so if manifests can be constructed to fit within datagrams, it is an appropriate choice. (IPSec is similar-worth adding as an option?).

This option provides no native redundancy or retransmission, but packet digests can be repeated in different manifests to provide some resilience to loss. Lost manifests that result in the loss of blocks of packet digests can be expensive, since they would make received data packets unauthenticatable. TBD: should we therefore not support this case?

3.4. DTLS + FECFRAME

DTLS for manifests that do not fit into single-packet payloads can still be delivered by using FECFRAME [RFC6363], particularly Reed-Solomon [RFC6865] or possibly Raptor [RFC6681]. This has some advantages compared to HTTPS because of the absence of HOL-blocking, while providing for tunable redundancy. This has some advantages relative to DTLS because of overhead reduction and non-integer redundancy tunability (e.g. 1.5 becomes a viable redundancy factor).

TBD: define this method, possibly in another RFC.

4. Examples

TBD: walk through some examples as soon as we have a build running. Likely to need some touching up.

5. YANG Module

5.1. Tree Diagram

```
module: ietf-ambi
  augment /dorms:metadata/dorms:sender/dorms:group/dorms:udp-stream:
    +--rw manifest-stream* [id]
      +--rw id                               uint32
      +--rw manifest-transport*             inet:uri
      +--rw hash-algorithm                  ct:asymmetric-key-algorithm-t
      +--rw payload-type                    enumeration
      +--rw data-hold-time-ms?              uint32
      +--rw digest-hold-time-ms?           uint32
```

5.2. Module

```
<CODE BEGINS> file ietf-ambi@2019-09-26.yang
module ietf-ambi {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-ambi";
  prefix "ambi";

  import ietf-dorms {
    prefix "dorms";
    reference "I-D.jholland-mboned-dorms";
  }

  import ietf-inet-types {
    prefix "inet";
    reference "RFC6991 Section 4";
  }

  import ietf-crypto-types {
    prefix "ct";
    reference "draft-ietf-netconf-crypto-types";
  }

  organization "IETF";

  contact
    "Author:   Jake Holland
              <mailto:jholland@akamai.com>";

  description
    "Copyright (c) 2019 IETF Trust and the persons identified as
    authors of the code. All rights reserved."
```

Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Simplified BSD License set forth in Section 4.c of the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC XXXX (<https://www.rfc-editor.org/info/rfcXXXX>); see the RFC itself for full legal notices.

The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document are to be interpreted as described in BCP 14 (RFC 2119) (RFC 8174) when, and only when, they appear in all capitals, as shown here.

This module contains the definition for the AMBI data types. It provides metadata for authenticating SSM channels as an augmentation to DORMS.";

```
revision 2019-08-25 {
    description "Initial revision as an extension.";
    reference
        "";
}

augment
    "/dorms:metadata/dorms:sender/dorms:group/dorms:udp-stream" {
        description "Definition of the manifest stream providing
            integrity info for the data stream";

        list manifest-stream {
            key id;
            description "Definition of a manifest stream.";
            leaf id {
                type uint32;
                mandatory true;
                description "The Manifest ID referenced in a manifest.";
            }
            leaf-list manifest-transport {
                type inet:uri;
                description "A URI that provides a location for the
                    manifest stream";
            }
            leaf hash-algorithm {
                type ct:asymmetric-key-algorithm-t;
                mandatory true;
            }
        }
    }
```

```
        description
            "The hash algorithm for the packet hashes within
            manifests in this stream.";
    }
    leaf payload-type {
        type enumeration {
            enum udp {
                description "The hash includes only the UDP
                payload.";
            }
            enum ip {
                description "The hash includes the full IP
                payload.";
            }
        }
        mandatory true;
        description "The contents of the payload for the
        digest profile";
    }
    leaf data-hold-time-ms {
        type uint32;
        default 2000;
        description "The number of milliseconds to hold data
        packets waiting for a corresponding digest before
        discarding";
    }
    leaf digest-hold-time-ms {
        type uint32;
        default 10000;
        description "The number of milliseconds to hold packet
        digests waiting for a corresponding data packet
        before discarding";
    }
}
}
```

<CODE ENDS>

6. IANA Considerations

6.1. The YANG Module Names Registry

This document adds one YANG module to the "YANG Module Names" registry maintained at <<https://www.iana.org/assignments/yang-parameters>>. The following registrations are made, per the format in Section 14 of [RFC6020]:

```
name:      ietf-ambi
namespace: urn:ietf:params:xml:ns:yang:ietf-ambi
prefix:    ambi
reference:  I-D.draft-jholland-mboned-ambi
```

6.2. Media Type

TBD: Register 'application/ambi' according to advice from:
<https://www.iana.org/form/media-types>

TBD: check guidelines in <https://tools.ietf.org/html/rfc5226>

7. Security Considerations

7.1. Predictable Packets

Protocols that have predictable packets run the risk of offline attacks for hash collisions against those packets. When authenticating a protocol that might have predictable packets, it's RECOMMENDED to use a hash function secure against such attacks or to add content to the packets to make them unpredictable, such as an Authentication Header ([RFC4302]), or the addition of an ignored field with random content to the packet payload.

TBD: explain attack from generating malicious packets and then looking for collisions, as opposed to having to generate a collision on packet contents that include a sequence number and then hitting a match (especially expand on this if we do add Section 2.3.1.2).

TBD: follow the rest of the guidelines: <https://tools.ietf.org/html/rfc3552>

8. Acknowledgements

Many thanks to Daniel Franke, Eric Rescorla, and Christian Worm Mortensen for their very helpful suggestions.

9. References

9.1. Normative References

[I-D.draft-jholland-mboned-dorms-00]
Holland, J., "Discovery Of Restconf Metadata for Source-specific multicast", draft-jholland-mboned-dorms-00 (work in progress), August 2019.

- [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>>.
- [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>>.
- [RFC6363] Watson, M., Begen, A., and V. Roca, "Forward Error Correction (FEC) Framework", RFC 6363, DOI 10.17487/RFC6363, October 2011, <<https://www.rfc-editor.org/info/rfc6363>>.
- [RFC6681] Watson, M., Stockhammer, T., and M. Luby, "Raptor Forward Error Correction (FEC) Schemes for FECFRAME", RFC 6681, DOI 10.17487/RFC6681, August 2012, <<https://www.rfc-editor.org/info/rfc6681>>.
- [RFC6865] Roca, V., Cunche, M., Lacan, J., Bouabdallah, A., and K. Matsuzono, "Simple Reed-Solomon Forward Error Correction (FEC) Scheme for FECFRAME", RFC 6865, DOI 10.17487/RFC6865, February 2013, <<https://www.rfc-editor.org/info/rfc6865>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

9.2. Informative References

- [I-D.draft-krose-mboned-alta-01]
Rose, K. and J. Holland, "Asymmetric Loss-Tolerant Authentication", draft-krose-mboned-alta-01 (work in progress), July 2019.
- [I-D.ietf-tsvwg-udp-options]
Touch, J., "Transport Options for UDP", draft-ietf-tsvwg-udp-options-08 (work in progress), September 2019.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, DOI 10.17487/RFC3550, July 2003, <<https://www.rfc-editor.org/info/rfc3550>>.

- [RFC4082] Perrig, A., Song, D., Canetti, R., Tygar, J., and B. Briscoe, "Timed Efficient Stream Loss-Tolerant Authentication (TESLA): Multicast Source Authentication Transform Introduction", RFC 4082, DOI 10.17487/RFC4082, June 2005, <<https://www.rfc-editor.org/info/rfc4082>>.
- [RFC4302] Kent, S., "IP Authentication Header", RFC 4302, DOI 10.17487/RFC4302, December 2005, <<https://www.rfc-editor.org/info/rfc4302>>.
- [RFC4383] Baugher, M. and E. Carrara, "The Use of Timed Efficient Stream Loss-Tolerant Authentication (TESLA) in the Secure Real-time Transport Protocol (SRTP)", RFC 4383, DOI 10.17487/RFC4383, February 2006, <<https://www.rfc-editor.org/info/rfc4383>>.
- [RFC5776] Roca, V., Francillon, A., and S. Faurite, "Use of Timed Efficient Stream Loss-Tolerant Authentication (TESLA) in the Asynchronous Layered Coding (ALC) and NACK-Oriented Reliable Multicast (NORM) Protocols", RFC 5776, DOI 10.17487/RFC5776, April 2010, <<https://www.rfc-editor.org/info/rfc5776>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.

Authors' Addresses

Jake Holland
Akamai Technologies, Inc.
150 Broadway
Cambridge, MA 02144
United States of America

Email: jakeholland.net@gmail.com

Kyle Rose
Akamai Technologies, Inc.
150 Broadway
Cambridge, MA 02144
United States of America

Email: krose@krose.org

Mboned
Internet-Draft
Intended status: Standards Track
Expires: March 29, 2020

J. Holland
Akamai Technologies, Inc.
September 26, 2019

Circuit Breaker Assisted Congestion Control
draft-jholland-mboned-cbacc-00

Abstract

This document specifies Circuit Breaker Assisted Congestion Control (CBACC). CBACC enables fast-trip Circuit Breakers by publishing rate metadata about multicast channels from senders to intermediate network nodes or receivers. The circuit breaker behavior is defined as a supplement to receiver driven congestion control systems, to preserve network health if receivers subscribe to a volume of traffic that exceeds capacity policies or capability for a network or receiver.

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 <https://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 March 29, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

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

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Background and Terminology	3
2. Circuit Breaker Behavior	4
2.1. Functional Components	4
2.1.1. Ingress Meter	4
2.1.2. Egress Meter	4
2.1.3. Communication Method	5
2.1.4. Measurement Function	5
2.1.5. Trigger Function	6
2.1.6. Reaction	7
2.1.7. Feedback Control Mechanism	7
2.2. States	7
2.2.1. Interface State	7
2.2.2. Flow State	8
2.3. Implementation Design Considerations	8
2.3.1. Oversubscription Thresholds	8
2.3.2. Fairness Functions	9
3. YANG Module	9
3.1. Tree Diagram	9
3.2. Module	9
4. IANA Considerations	11
4.1. YANG Module Names Registry	11
5. Security Considerations	11
5.1. Metadata Security	11
5.2. Denial of Service	11
5.2.1. State Overload	11
6. Acknowledgements	12
7. References	12
7.1. Normative References	12
7.2. Informative References	12
Appendix A. Overjoining	14
Author's Address	15

1. Introduction

This document defines Circuit Breaker Assisted Congestion Control (CBACC). CBACC defines a Network Transport Circuit Breaker (CB), as described by [RFC8084].

The CB behavior defined in this document uses bit-rate metadata about multicast data streams, coupled with policy, capacity, and load information at a network node, to prune multicast channels so that

the node's aggregate capacity is not exceeded by the subscribed channels.

To communicate the required metadata, this document defines a YANG [RFC7950] module that augments the DORMS [I-D.draft-jholland-mboned-dorms-00] YANG module. DORMS provides a mechanism for senders to publish metadata about the multicast streams they're sending through a RESTCONF service, so that receivers or forwarding nodes can discover and consume the metadata with a set of standard methods. The metadata MAY be communicated to receivers or forwarding nodes by some other method, but the definition of any alternative methods is out of scope for this document.

The CB behavior defined in this document matches the description provided in Section 3.2.3 of [RFC8084] of a unidirectional CB over a controlled path. The control messages from that description are composed of the messages containing the metadata required for operation of the CB.

CBACC is designed to supplement protocols that use multicast IP and rely on well-behaved receivers to achieve congestion control. Examples of congestion control systems fitting this description include [PLM], [RLM], [RLC], [FLID-DL], [SMCC], and WEBRC [RFC3738].

CBACC addresses a problem with "overjoining" by untrusted receivers.

In an overjoining condition, receivers (either malicious, misconfigured, or with implementation errors) subscribe to multicast channels but do not respond appropriately to congestion. When sufficient multicast traffic is available for subscription by such receivers, this can overload any network.

The overjoining problem is relevant to misbehaving receivers for both receiver-driven and feedback-driven congestion control strategies, as described in Section 4.1 of [RFC8085].

Overjoining attacks and the challenges they present are discussed in more detail in Appendix A.

CBACC offers a solution for the recommendation in Section 4 of [RFC8085] that circuit breaker solutions be used even where congestion control is optional.

1.1. Background and Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP

14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Circuit Breaker Behavior

2.1. Functional Components

This section maps the functional components described in Section 3.1 of [RFC8084] to the operational components of the CBACC CB defined by this document.

2.1.1. Ingress Meter

The metadata provides an ingress meter in the form of an advertised maximum data bit-rate, namely the "max-bits-per-second" field in the YANG model in Section 3. This is a self-report by the sender about the maximum amount of traffic a sender will send within the interval given by the "data-rate-window" field, which is the measurement interval.

The sender MUST NOT send more data for a data stream than the amount of data implied by its advertised data rate within any measurement window, and it's RECOMMENDED for the sender to provide some margin to account for forwarding bursts. If an egress node observes a higher data rate within any measurement window, it MAY circuit-break that flow immediately.

2.1.2. Egress Meter

The node implementing the CB behavior has access to several pieces of information that can be used as relevant egress metrics:

1. Physical capacity limits on each interface.
2. Configured capacity limits for multicast traffic for each interface, if any.
3. The observed received data rates of subscribed multicast channels with CBACC metadata.
4. The observed received data rates of subscribed multicast channels without CBACC metadata.
5. The observed received data rates of competing non-multicast traffic.
6. The loss rate for subscribed multicast channels, when available. The loss rate is only sometimes observable at an egress node; for

example, when using AMBI [I-D.draft-jholland-mboned-ambi-03], or when the data stream carries a protocol that is known to the egress node by some out of band means, and whose traffic can be monitored for loss. When available, the loss rates may be used.

Note that any on-path router can be considered an egress node for purposes of this CB, even though it may be forwarding traffic downstream, and even though other egress points may also be operating a downstream CB that covers the same data stream. Components in the receiving devices, such as an operating system or browser can also act as an egress node, as can a receiving application.

2.1.3. Communication Method

CBACC operates at an egress node, so the egress metrics in Section 2.1.2 are available through system calls, or by communication with various locally deployable system monitoring applications. Any suitable application that provides the necessary egress meter is appropriate.

The communication path defined in this document for the information from the ingress meter is the use of DORMS [I-D.draft-jholland-mboned-dorms-00]. Other methods MAY be used as well, but are out of scope for this document.

2.1.4. Measurement Function

The measurement function maintains a few values for each interface, computed from the egress and ingress meter values:

1. The aggregate advertised maximum bit-rate capacity consumed by CBACC data streams. This is the sum of the max-bit-rate values in the CBACC metadata for all data streams subscribed through an interface
2. An oversubscription threshold for each interface. The oversubscription threshold will be determined differently for CBs in different contexts. In some network devices, it might be as simple as an administratively configured absolute value or proportion of an interface's capacity. For other situations, like a CB operating in a context with loss visibility, it could be a dynamically changing value that grows when data streams are successfully subscribed and receiving data without loss, and shrinks as loss is observed across subscribed data streams. The oversubscription threshold calculation could also incorporate other information like out-of-band path capacity measurements with [PathChirp], if available.

This document covers some non-normative examples of valid oversubscription threshold functions in Section 2.3.1, but in general, the oversubscription threshold is the primary parameter that different CBs in different contexts can tune to provide the safety guarantees necessary for their context.

2.1.5. Trigger Function

The trigger function fires when the aggregate advertised maximum bit-rate exceeds the oversubscription threshold for any interface.

When oversubscribed, the trigger function changes the states of subscribed channels to "blocked" until the aggregate subscribed bit-rate is below the oversubscription threshold again.

2.1.5.1. Fairness and Inter-flow Ordering

The trigger function orders the monitored flows according to a fairness function, and blocks flows in order as needed to ensure that only a safe level of bandwidth can be consumed by subscribed flows. The fairness function can be different for CBs in different contexts.

Flows from a single sender MUST be ordered according to their priority field from the CBACC metadata when compared with each other. Between-sender flows and flows from the same sender with the same priority are ordered according to the fairness function. Where flows from the same sender have a priority order that conflicts with the ordering the fairness function would use, it's appropriate to treat those out of order flows from the sender as an aggregate flow for between-sender flow comparisons. (TBD: the aggregation algorithm probably needs more explaining and good examples.)

A CB implementation SHOULD provide mechanisms for administrative controls to configure explicit biases, as this may be necessary to support Service Level Agreements for specific events or providers, or to blacklist or de-prioritize channels with historically known misbehavior.

Subject to the above constraints, where possible the default fairness behavior SHOULD favor streams with many receivers over streams with few receivers, and streams with a low bit-rate over streams with a high bit-rate. For example, when receiver count is known, a good fairness metric is max-bandwidth divided by receiver-count. (Receiver count in some networks can be known through technologies such as the experimental PIM extension for population count described in [RFC6807], or other custom signaling methods.)

An overview of some other approaches to appropriate fairness metrics is given in Section 2.3 of [RFC5166].

2.1.6. Reaction

When the trigger function fires and a subscribed channel becomes blocked, the reaction depends on whether it's an upstream interface or a downstream interface.

If a channel is blocked on one downstream interface, it may still be unblocked on other downstream interfaces. When this is the case, traffic is simply not forwarded along blocked interfaces, even though clients might still be joined.

When a channel is blocked on all downstream interfaces, or when the upstream interface is oversubscribed, the channel is pruned so that data no longer arrives from the network on the upstream interface, by a PIM prune (Section 3.5 of [RFC7761]), or a "leave" operation with IGMP, MLD, or another multicast signaling mechanism.

Once initially circuit-broken, a flow SHOULD remain circuit-broken for no less than 3 minutes, even if space clears up, to ensure downstream subscriptions will notice and respond. (3 minutes is chosen to exceed the default maximum lifetime of 2 minutes that can occur if an IGMP responder suddenly stops operation, and ceases responding to IGMP queries with membership reports.)

When enough capacity is available for a circuit-broken stream to be unblocked and the circuit-breaker hold-down time is expired, the flows SHOULD be unblocked according to the priority order.

2.1.7. Feedback Control Mechanism

The metadata should be refreshed as needed to maintain up to date values. When using DORMS and RESTCONF, the HTTP Cache Control headers provide valid refresh time properties from the server, and SHOULD be used if present. If No-Cache is used, the default refresh timing SHOULD be 30 seconds plus a random value between 0 and 10 seconds.

2.2. States

2.2.1. Interface State

A CB holds the following state for each interface, for both the inbound and outbound directions on that interface:

- o aggregate bandwidth: The sum of the bandwidths of all non-circuit-broken CBACC flows which transit this interface in this direction.
- o bandwidth limit: The maximum aggregate CBACC advertised bandwidth allowed, not including circuit-broken flows.

When reducing the bandwidth limit due to congestion, the circuit breaker SHOULD NOT reduce the limit by more than half its value in 10 seconds, and SHOULD use a smoothing function to reduce the limit gradually over time.

It is RECOMMENDED that no more than half the capacity for a link be allocated to CBACC flows if the link might be shared with TCP or other traffic that is responsive to congestion.

2.2.2. Flow State

Data streams with CBACC metadata have a state for the upstream interface through which the stream is joined:

- o 'subscribed' Indicates that the circuit breaker is subscribed upstream to the flow and forwarding packets through zero or more egress interfaces.
- o 'pruned' Indicates that the flow has been circuit-broken. A request to unsubscribe from the flow has been sent upstream, e.g. a PIM prune (Section 3.5 of [RFC7761]) or a "leave" operation via IGMP, MLD, or another group membership management mechanism.

Data streams also have a per-interface state for downstream interfaces with subscribers, where the data is being forwarded. It's one of:

- o 'forwarding' Indicates that the flow is a non-circuit-broken flow in steady state, forwarding packets downstream.
- o 'blocked' Indicates that data packets for this flow are NOT forwarded downstream via this interface.

2.3. Implementation Design Considerations

2.3.1. Oversubscription Thresholds

TBD.

2.3.2. Fairness Functions

TBD.

3. YANG Module

3.1. Tree Diagram

```
module: ietf-cbacc
  augment /dorms:metadata/dorms:sender/dorms:group/dorms:udp-stream:
    +--rw cbacc!
      +--rw max-bits-per-second      uint32
      +--rw max-mss?                 uint16
      +--rw data-rate-window?        uint32
      +--rw priority?                uint16
```

3.2. Module

```
<CODE BEGINS> file ietf-cbacc@2019-09-26.yang
module ietf-cbacc {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-cbacc";
  prefix "ambi";

  import ietf-dorms {
    prefix "dorms";
    reference "I-D.jholland-mboned-dorms";
  }

  organization "IETF";

  contact
    "Author:   Jake Holland
              <mailto:jholland@akamai.com>";

  description
    "This module contains the definition for the AMBI anchor
    message data type.

    Copyright (c) 2019 IETF Trust and the persons identified as
    authors of the code. All rights reserved.

    Redistribution and use in source and binary forms, with or
    without modification, is permitted pursuant to, and subject
    to the license terms contained in, the Simplified BSD License
```

set forth in Section 4.c of the IETF Trust's Legal Provisions
Relating to IETF Documents
(<http://trustee.ietf.org/license-info>).

This version of this YANG module is part of
draft-jholland-mboned-cbacc, [TBD: change]
see the internet draft itself for full legal notices.";

```
revision 2019-07-31 {
  description "Initial revision as an extension.";
  reference
    "";
}

augment
  "/dorms:metadata/dorms:sender/dorms:group/dorms:udp-stream" {
    description "Definition of the manifest stream providing
      integrity info for the data stream";

    container cbacc {
      presence "cbacc-enabled flow";
      description "Information to enable fast-trip circuit breakers";
      leaf max-bits-per-second {
        type uint32;
        mandatory true;
        description "Maximum bitrate for this stream, in Kilobits
          of IP packet data (including headers) of native
          multicast traffic per second";
      }
      leaf max-mss {
        type uint16;
        default 1400;
        description "Maximum payload size, in bytes";
      }
      leaf data-rate-window {
        type uint32;
        default 2000;
        description "Time window over which data rate is guaranteed,
          in milliseconds.";
        /* TBD: range limits? */
      }
      leaf priority {
        type uint16;
        default 256;
        description "The relative preference level for keeping this
          flow compared to other flows from this sender (higher
          value is more preferred to keep)";
      }
    }
  }
```

```
    }  
  }  
}
```

<CODE ENDS>

4. IANA Considerations

4.1. YANG Module Names Registry

This document adds one YANG module to the "YANG Module Names" registry maintained at <https://www.iana.org/assignments/yang-parameters>. The following registrations are made, per the format in Section 14 of [RFC6020]:

```
name:      ietf-cbacc  
namespace: urn:ietf:params:xml:ns:yang:ietf-cbacc  
prefix:    cbacc  
reference: I-D.draft-jholland-mboned-cbacc
```

5. Security Considerations

5.1. Metadata Security

Be sure to authenticate the metadata. See DORMS security considerations, and don't accept unauthenticated metadata if using an alternative means.

5.2. Denial of Service

5.2.1. State Overload

Since CBACC flows require state, it may be possible for a set of receivers and/or senders, possibly acting in concert, to generate many flows in an attempt to overflow the circuit breakers' state tables.

It is permissible for a network node to behave as a CBACC circuit breaker for some CBACC flows while treating other CBACC flows as non-CBACC, as part of a load balancing strategy for the network as a whole, or simply as defense against this concern when the number of monitored flows exceeds some threshold.

The same techniques described in Section 3.1 of [RFC4609] can be used to help mitigate this attack, for much the same reasons. It is RECOMMENDED that network operators implement measures to mitigate such attacks.

6. Acknowledgements

Many thanks to Devin Anderson, Ben Kaduk, Cheng Jin, Scott Brown, Miroslav Ponec, Bob Briscoe, Lenny Giuliani, and Christian Worm Mortensen for their thoughtful comments and contributions.

7. References

7.1. Normative References

- [I-D.draft-jholland-mboned-ambi-03]
Holland, J. and K. Rose, "Asymmetric Manifest Based Integrity", draft-jholland-mboned-ambi-03 (work in progress), September 2019.
- [I-D.draft-jholland-mboned-dorms-00]
Holland, J., "Discovery Of Restconf Metadata for Source-specific multicast", draft-jholland-mboned-dorms-00 (work in progress), August 2019.
- [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>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8084] Fairhurst, G., "Network Transport Circuit Breakers", BCP 208, RFC 8084, DOI 10.17487/RFC8084, March 2017, <<https://www.rfc-editor.org/info/rfc8084>>.
- [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>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

7.2. Informative References

- [FLID-DL] Byers, J., Horn, G., Luby, M., Mitzenmacher, M., Shaver, W., and IEEE, "FLID-DL: congestion control for layered multicast", DOI 10.1109/JSAC.2002.803998, n.d., <<https://ieeexplore.ieee.org/document/1038584>>.

- [PathChirp] Ribeiro, V., Riedi, R., Baraniuk, R., Navratil, J., Cottrell, L., Department of Electrical and Computer Engineering Rice University, and SLAC/SCS-Network Monitoring, Stanford University, "pathChirp: Efficient Available Bandwidth Estimation for Network Paths", 2003.
- [PLM] Biersack, Institut EURECOM, A., "PLM: Fast Convergence for Cumulative Layered Multicast Transmission Schemes", 1999.
- [RFC3738] Luby, M. and V. Goyal, "Wave and Equation Based Rate Control (WEBRC) Building Block", RFC 3738, DOI 10.17487/RFC3738, April 2004, <<https://www.rfc-editor.org/info/rfc3738>>.
- [RFC4609] Savola, P., Lehtonen, R., and D. Meyer, "Protocol Independent Multicast - Sparse Mode (PIM-SM) Multicast Routing Security Issues and Enhancements", RFC 4609, DOI 10.17487/RFC4609, October 2006, <<https://www.rfc-editor.org/info/rfc4609>>.
- [RFC5166] Floyd, S., Ed., "Metrics for the Evaluation of Congestion Control Mechanisms", RFC 5166, DOI 10.17487/RFC5166, March 2008, <<https://www.rfc-editor.org/info/rfc5166>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.
- [RFC6807] Farinacci, D., Shepherd, G., Venaas, S., and Y. Cai, "Population Count Extensions to Protocol Independent Multicast (PIM)", RFC 6807, DOI 10.17487/RFC6807, December 2012, <<https://www.rfc-editor.org/info/rfc6807>>.
- [RFC7761] Fenner, B., Handley, M., Holbrook, H., Kouvelas, I., Parekh, R., Zhang, Z., and L. Zheng, "Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised)", STD 83, RFC 7761, DOI 10.17487/RFC7761, March 2016, <<https://www.rfc-editor.org/info/rfc7761>>.
- [RLC] Rizzo, L., Vicisano, L., and J. Crowcroft, "The RLC multicast congestion control algorithm", 1999.
- [RLM] McCanne, S., Jacobson, V., Vetterli, M., University of California, Berkeley, and Lawrence Berkeley National Laboratory, "Receiver-driven Layered Multicast", 1995.

[SMCC] Kwon, G., Byers, J., and Computer Science Department,
Boston University, "Smooth Multirate Multicast Congestion
Control", 2002.

Appendix A. Overjoining

[RFC8085] describes several remedies for unicast congestion control under UDP, even though UDP does not itself provide congestion control. In general, any network node under congestion could in theory collect evidence that a unicast flow's sending rate is not responding to congestion, and would then be justified in circuit-breaking it.

With multicast IP, the situation is different, especially in the presence of malicious receivers. A well-behaved sender using a receiver-controlled congestion scheme such as WEBRC does not reduce its send rate in response to congestion, instead relying on receivers to leave the appropriate multicast groups.

This leads to a situation where, when a network accepts inter-domain multicast traffic, as long as there are senders somewhere in the world with aggregate bandwidth that exceeds a network's capacity, receivers in that network can join the flows and overflow the network capacity. A receiver controlled by an attacker could do this at the IGMP/MLD level without running the application layer protocol that participates in the receiver-controlled congestion control.

A network might be able to detect and defend against the most naive version of such an attack by blocking end users that try to join too many flows at once. However, an attacker can achieve the same effect by joining a few high-bandwidth flows, if those exist anywhere, and an attacker that controls a few machines in a network can coordinate the receivers so they join disjoint sets of non-responsive sending flows.

This scenario will produce congestion in a middle node in the network that can't be easily detected at the edge where the IGMP/MLD join is accepted. Thus, an attacker with a small set of machines in a target network can always trip a circuit breaker if present, or can induce excessive congestion among the bandwidth allocated to multicast. This problem gets worse as more multicast flows become available.

Although the same can apply to non-responsive unicast traffic, network operators can assume that non-responsive sending flows are in violation of congestion control best practices, and can therefore cut off flows associated with the misbehaving senders. By contrast, non-responsive multicast senders are likely to be well-behaved participants in receiver-controlled congestion control schemes.

However, receiver controlled congestion control schemes also show the most promise for efficient massive scale content distribution via multicast, provided network health can be ensured. Therefore, mechanisms to mitigate overjoining attacks while still permitting receiver-controlled congestion control are necessary.

Author's Address

Jake Holland
Akamai Technologies, Inc.
150 Broadway
Cambridge, MA 02144
United States of America

Email: jakeholland.net@gmail.com

Mboned
Internet-Draft
Intended status: Standards Track
Expires: March 29, 2020

J. Holland
Akamai Technologies, Inc.
September 26, 2019

Discovery Of Restconf Metadata for Source-specific multicast
draft-jholland-mboned-dorms-01

Abstract

This document defines DORMS (Discovery Of Restconf Metadata for Source-specific multicast), a method to discover and retrieve extensible metadata about source-specific multicast channels using RESTCONF. The reverse IP DNS zone for a multicast sender's IP address is configured to use SRV resource records to advertise the hostname of a RESTCONF server that publishes metadata according to a new YANG module with support for extensions. A new service name and the new YANG module are defined.

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 <https://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 March 29, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

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

include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
1.1. Background	3
1.2. Terminology	3
2. Discovery and Metadata Retrieval	4
2.1. DNS Bootstrap	4
2.2. RESTCONF Bootstrap	5
2.2.1. Root Resource Discovery	5
2.2.2. Yang Library Version	6
2.2.3. Yang Library Contents	6
2.2.4. Metadata Retrieval	7
2.2.5. Cross Origin Resource Sharing (CORS)	8
3. Scalability Considerations	9
3.1. Provisioning	9
3.2. Data Scoping	9
4. YANG Model	9
4.1. Yang Tree	10
4.2. Yang Module	10
5. Privacy Considerations	12
6. IANA Considerations	12
6.1. The YANG Module Names Registry	12
6.2. The Service Name and Transport Protocol Port Number Registry	13
7. Security Considerations	13
7.1. Secure Communications	13
7.2. Exposure of Metadata	13
7.3. DNS Bootstrapping	14
8. Acknowledgements	14
9. References	14
9.1. Normative References	14
9.2. Informative References	15
Author's Address	17

1. Introduction

This document defines DORMS (Discovery Of Restconf Metadata for Source-specific multicast).

A DORMS service is a RESTCONF [RFC8040] service that provides read access to data in the "ietf-dorms" YANG [RFC7950] model defined in Section 4. This model, along with optional extensions defined in other documents, provide an extensible set of information about multicast data streams.

This document defines the "dorms" service name for use with the SRV DNS Resource Record (RR) type [RFC2782]. A sender offering a DORMS service to publish metadata SHOULD configure at least one SRV RR for the "_dorms._tcp" subdomain in the reverse IP DNS zone for the source IP of its multicast channel to advertise a hostname for a DORMS server that can provide metadata for the sender's source-specific multicast traffic. Doing so enables receivers and middleboxes to discover and query a DORMS server as described in Section 2.

The goal is to provide an extensible framework for attaching information necessary for the correct processing of multicast data channels, both for middle boxes forwarding the traffic, and for receivers subscribing to traffic (hereafter called "clients").

1.1. Background

The reader is assumed to be familiar with the basic DNS concepts described in [RFC1034], [RFC1035], and the subsequent documents that update them, as well as the use of the SRV Resource Record type as described in [RFC2782].

The reader is also assumed to be familiar with the concepts and terminology regarding source-specific multicast as described in [RFC4607] and the use of IGMPv3 [RFC3376] and MLDv2 [RFC3810] for group management of source-specific multicast channels, as described in [RFC4604].

The reader is also assumed to be familiar with the concepts and terminology for RESTCONF [RFC8040] and YANG [RFC7950].

1.2. Terminology

Term	Definition
(S,G)	A source-specific multicast channel, as described in [RFC4607]. A pair of IP addresses with a source host IP and destination group IP.
RR	A DNS Resource Record, as described in [RFC1034]
RRType	A DNS Resource Record Type, as described in [RFC1034]
SSM	Source-specific multicast, as described in [RFC4607]

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and

"OPTIONAL" in this document are to be interpreted as described in [RFC2119] and [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Discovery and Metadata Retrieval

A client that needs metadata about a (S,G) MAY attempt to discover metadata for the (S,G) using the mechanisms defined here, and MAY use the metadata received to manage the forwarding or processing of the packets in the channel.

2.1. DNS Bootstrap

The DNS Bootstrap step is how a client discovers an appropriate RESTCONF server, given the source address of an (S,G). Use of the DNS Bootstrap is OPTIONAL for clients with an alternate method of obtaining a RESTCONF hostname for a DORMS server with metadata for an (S,G).

This mechanism only works for source-specific multicast (SSM) channels. The source address of the (S,G) is reversed and used as an index into one of the reverse mapping trees (in-addr.arpa for IPv4, as described in Section 3.5 of [RFC1035], or ip6.arpa for IPv6, as described in Section 2.5 of [RFC3596]).

When a receiver or middle box needs metadata for an (S,G), for example when handling a new join for that (S,G) and looking up authentication methods available, a receiver or middlebox can issue a DNS query for a SRV RR using the "dorms" service name with the domain from the reverse mapping tree, combining them as described in [RFC2782].

For example, while handling a join for (203.0.113.15, 232.1.1.1), a receiver would perform a DNS query for the SRV RRTYPE for the domain:

`_dorms._tcp.15.113.0.203.in-addr.arpa.`

The DNS response for this domain might return a record such as:

`SRV 0 1 443 dorms-restconf.example.com.`

This response informs the receiver that a DORMS server SHOULD be reachable at dorms-restconf.example.com on port 443. Multiple SRV records are handled as described by [RFC2782].

A sender providing DORMS discovery SHOULD publish at least one SRV record in the reverse DNS zone for each source address of the multicast channels it is sending, in order to advertise the hostname

of the DORMS server to receivers and middle boxes. The DORMS servers advertised SHOULD be configured with metadata for all the groups sent from the same source IP address that have metadata published with DORMS.

2.2. RESTCONF Bootstrap

Once a DORMS host has been chosen (whether via an SRV RR from a DNS response or via some other method), RESTCONF provides all the information necessary to determine the versions and url paths for metadata from the server. A walkthrough is provided here for a sequence of example requests and responses from a receiver connecting to a new DORMS server.

2.2.1. Root Resource Discovery

As described in Section 3.1 of [RFC8040] and [RFC6415], the RESTCONF server provides the link to the RESTCONF api entry point via the `"/.well-known/host-meta"` or `"/.well-known/host-meta.json"` resource.

Example:

The receiver might send:

```
GET /.well-known/host-meta.json HTTP/1.1
Host: dorms-restconf.example.com
Accept: application/json
```

The server might respond as follows:

```
HTTP/1.1 200 OK
Date: Tue, 27 Aug 2019 20:56:00 GMT
Server: example-server
Cache-Control: no-cache
Content-Type: application/json
```

```
{
  "links":[
    {
      "rel":"restconf",
      "href":"/top/restconf"
    }
  ]
}
```

2.2.2. Yang Library Version

As described in Section 3.3.3 of [RFC8040], the yang-library-version leaf is required by RESTCONF, and can be used to determine the schema of the ietf-yang-library module:

Example:

The receiver might send:

```
GET /top/restconf/yang-library-version HTTP/1.1
Host: dorms-restconf.example.com
Accept: application/yang-data+json
```

The server might respond as follows:

```
HTTP/1.1 200 OK
Date: Tue, 27 Aug 2019 20:56:01 GMT
Server: example-server
Cache-Control: no-cache
Content-Type: application/yang-data+json

{
  "ietf-restconf:yang-library-version": "2016-06-21"
}
```

TBD: We might need a method for learning a specific restconf server or resource path that supports a version the client knows how to use, in the case the client is older than the server after a new yang-library version is released... Can this be just retry with a hold-down on specific hostnames, so that you can find a lower priority older server from the SRV records, or is signaling that can find or negotiate an explicit version as part of the lookup going to be necessary? -jake 2019-08-26

2.2.3. Yang Library Contents

After checking that the version of the yang-library module will be understood by the receiver, the client can check that the desired metadata module is available on the DORMS server by fetching the module-state resource from the ietf-yang-library module.

Example:

The receiver might send:

```
GET /top/restconf/data/ietf-yang-library:modules-state/\
    module=ietf-dorms,2016-08-15
Host: dorms-restconf.example.com
Accept: application/yang-data+json
```

The server might respond as follows:

```
HTTP/1.1 200 OK
Date: Tue, 27 Aug 2019 20:56:02 GMT
Server: example-server
Cache-Control: no-cache
Content-Type: application/yang-data+json

{
  "ietf-yang-library:module": [
    {
      "conformance-type": "implement",
      "name": "ietf-dorms",
      "namespace": "urn:ietf:params:xml:ns:yang:ietf-dorms",
      "revision": "2019-08-25",
      "schema":
        "https://example.com/yang/ietf-dorms@2019-08-25.yang"
    }
  ]
}
```

Other modules required or desired by the client also can be checked in a similar way, or the full set of available modules can be retrieved by not providing a key for the "module" list.

2.2.4. Metadata Retrieval

Once the expected DORMS version is confirmed, the client can retrieve the metadata specific to the desired (S,G).

Example:

The receiver might send:

```
GET /top/restconf/data/ietf-dorms:metadata/\
    sender=203.0.113.15/group=232.1.1.1
Host: dorms-restconf.example.com
Accept: application/yang-data+json
```

The server might respond as follows:

```
HTTP/1.1 200 OK
Date: Tue, 27 Aug 2019 20:56:02 GMT
Server: example-server
Cache-Control: no-cache
Content-Type: application/yang-data+json
```

```
{
  "ietf-dorms:group": [
    {
      "group-address": "232.1.1.1",
      "udp-stream": [
        {
          "port": "5001"
        }
      ]
    }
  ]
}
```

Note that when other modules are installed on the DORMS server that extend the ietf-dorms module, other fields MAY appear inside the response. This is the primary mechanism for providing extensible metadata for an (S,G), so clients SHOULD ignore fields they do not understand.

As mentioned in Section 3.2, most clients SHOULD use data resource identifiers in the request URI as in the above example, in order to retrieve metadata for only the targeted (S,G)s.

2.2.5. Cross Origin Resource Sharing (CORS)

It is RECOMMENDED that DORMS servers use the Access-Control-Allow-Origin header field, as specified by [W3C.REC-cors-20140116], and that they respond appropriately to Preflight requests.

Providing '*' for the allowed origins exposes the DORMS-based metadata to all web pages. When access to the metadata is used as a prerequisite to permitting the joining of the multicast flows, this would permit scripts from arbitrary web pages to issue joins for the multicast flows, which could allow e.g. malicious advertisements to participate in overjoining attacks (see Appendix A of [I-D.draft-jholland-cb-assisted-cc-01]) using multicast flows not controlled by the ad's senders. Therefore the use of '*' for allowed origins is NOT RECOMMENDED. (TBD: this probably deserves a security considerations section.)

3. Scalability Considerations

3.1. Provisioning

In contrast to many common RESTCONF deployments that are intended to provide configuration management for a service to a narrow set of authenticated administrators, DORMS servers often provide read-only metadata for public access, or for a very large set of end receivers, since it provides metadata in support of multicast data streams and multicast can scale to very large audiences.

Operators are advised to provision the DORMS service in a way that will scale appropriately to the size of the expected audience. Specific advice on such scaling is out of scope for this document, but some of the mechanisms outlined in [RFC3040] or other online resources might be useful, depending on the expected number of receivers.

3.2. Data Scoping

In the absence of contextual information, clients SHOULD issue narrowed requests for DORMS resources by following the format from Section 3.5.3 of [RFC8040] to encode data resource identifiers in the request URI. This avoids downloading excessive data, since the DORMS server may provide metadata for many (S,G)s, possibly from many different senders.

However, clients MAY use heuristics or out of band information about the service to issue requests for (S,G) metadata narrowed only by the source-address, or not narrowed at all. Depending on the request patterns and the contents of the data store, this may result in fewer round trips or less overhead, and can therefore be helpful behavior for scaling purposes. Servers MAY restrict or throttle client access based on the client certificate presented (if any), or based on heuristics that take note of client request patterns.

A complete description of the heuristics for clients and servers to meet their scalability goals is out of scope for this document.

4. YANG Model

The primary purpose of the YANG model defined here is to serve as a scaffold for the more useful metadata that will extend it. Currently known use cases include providing authentication information and bit-rate information for use by receivers and middle boxes, but more use cases are anticipated.

4.1. Yang Tree

```
module: ietf-dorms
  +--rw metadata
    +--rw sender* [source-address]
      +--rw source-address    inet:ip-address
    +--rw group* [group-address]
      +--rw group-address    rt-types:ip-multicast-group-address
    +--rw udp-stream* [port]
      +--rw port            inet:port-number
```

DORMS Tree Diagram

4.2. Yang Module

```
<CODE BEGINS> file ietf-dorms@2019-09-26.yang
module ietf-dorms {
  yang-version 1.1;

  namespace "urn:ietf:params:xml:ns:yang:ietf-dorms";
  prefix "dorms";

  import ietf-inet-types {
    prefix "inet";
    reference "RFC 6991 Section 4";
  }

  import ietf-routing-types {
    prefix "rt-types";
    reference "RFC 8294";
  }

  organization "IETF";

  contact
    "Author:   Jake Holland
              <mailto:jholland@akamai.com>";

  description
    "Copyright (c) 2019 IETF Trust and the persons identified as
    authors of the code.  All rights reserved.

    Redistribution and use in source and binary forms, with or
    without modification, is permitted pursuant to, and subject to
    the license terms contained in, the Simplified BSD License set
    forth in Section 4.c of the IETF Trust's Legal Provisions
```

Relating to IETF Documents
(<https://trustee.ietf.org/license-info>).

This version of this YANG module is part of RFC XXXX
(<https://www.rfc-editor.org/info/rfcXXXX>); see the RFC itself
for full legal notices.

The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED', 'MAY', and 'OPTIONAL' in this document are to be interpreted as described in BCP 14 (RFC 2119) (RFC 8174) when, and only when, they appear in all capitals, as shown here.

This module contains the definition for the DORMS data type.
It provides out of band metadata about SSM channels.";

```
revision 2019-08-25 {
  description "Initial revision.";
  reference
    "";
    // "I-D.draft-jholland-mboned-dorms";
}

container metadata {
  description "Metadata scaffold for source-specific multicast
    channels.";
  list sender {
    key source-address;
    description "Sender for DORMS";

    leaf source-address {
      type inet:ip-address;
      mandatory true;
      description
        "The source IP address of a multicast sender.";
    }

    list group {
      key group-address;
      description "Metadata for a DORMS (S,G).";

      leaf group-address {
        type rt-types:ip-multicast-group-address;
        mandatory true;
        description "The group IP address for an (S,G).";
      }

      list udp-stream {
```

```

    key "port";
    description
        "Metadata for UDP traffic on a specific port.";
    leaf port {
        type inet:port-number;
        mandatory true;
        description
            "The UDP port of a data stream in an (S,G).";
    }
}
}
}
}
}
<CODE ENDS>

```

5. Privacy Considerations

In the typical case, the mechanisms defined in this document provide a standardized way to discover information that is already available in other ways.

However, depending on the metadata provided by the server, observers may be able to more easily associate traffic from an (S,G) with the content contained within the (S,G). At the subscriber edge of a multicast-capable network, where the network operator has the capability to localize a IGMP [RFC3376] or MLD [RFC3810] channel subscription to a specific user or location by MAC address or source IP address, the structured publishing of metadata may make it easier to automate collection of data about the content a receiver is consuming.

RESTCONF communication will expose to the controller of the RESTCONF server information about the consumers of content that is not otherwise available from multicast subscription to a channel, such as the remote IP address and timing information.

6. IANA Considerations

6.1. The YANG Module Names Registry

This document adds one YANG module to the "YANG Module Names" registry maintained at <https://www.iana.org/assignments/yang-parameters>. The following registrations are made, per the format in Section 14 of [RFC6020]:

```
name:      ietf-dorms
namespace: urn:ietf:params:xml:ns:yang:ietf-dorms
prefix:    dorms
reference:  I-D.draft-jholland-mboned-dorms
```

6.2. The Service Name and Transport Protocol Port Number Registry

This document adds one service name to the "Service Name and Transport Protocol Port Number Registry" maintained at <https://www.iana.org/assignments/service-names-port-numbers>. The following registrations are made, per the format in Section 8.1.1 of [RFC6335]:

```
Service Name:      dorms
Transport Protocol(s): TCP
Assignee:          IESG <iesg@ietf.org>
Contact:           IETF Chair <chair@ietf.org>
Description:       This service name is used to construct the
                   SRV service label "_dorms" for discovering
                   DORMS servers.
Reference:         I-D.draft-jholland-mboned-dorms
Port Number:       N/A
Service Code:      N/A
Known Unauthorized Uses: N/A
Assignment Notes:  This protocol uses HTTPS as a substrate.
```

7. Security Considerations

7.1. Secure Communications

It is intended that security related metadata about the SSM channels will be delivered over the RESTCONF connection, and that information available from this connection can be used as a trust anchor.

The provisions of Section 2 of [RFC8040] provide secure communication requirements that are already required of DORMS servers, since they are RESTCONF servers. All RESTCONF requirements and security considerations remain in force for DORMS servers.

7.2. Exposure of Metadata

Although some DORMS servers MAY restrict access based on client identity, as described in Section 2.5 of [RFC8040], many DORMS servers will use the ietf-dorms YANG model to publish information without restriction, and even DORMS servers requiring client authentication will inherently, because of the purpose of DORMS, be providing the DORMS metadata to potentially many receivers.

Accordingly, future YANG modules that augment data paths under "ietf-dorms:metadata" MUST NOT include any sensitive data unsuitable for public dissemination in those data paths. Because of the possibility that scalable read-only access might be necessary to fulfill the scalability goals for a DORMS server, data under these paths MAY be cached or replicated by numerous external entities, so owners of such data SHOULD NOT assume it can be kept secret when provided by DORMS servers anywhere under the "ietf-dorms:metadata" path, even if they are authenticating clients.

7.3. DNS Bootstrapping

The DNS bootstrap phase relies on DNS for the reverse IP tree. When using DNS to discover a DORMS server's domain name, there must be a trust relationship between the end consumer of this resource record and the DNS server. This relationship may be end-to-end DNSSEC validation, a TSIG [RFC2845] or SIG(0) [RFC2931] channel to another secure source, a secure local channel on the host, DNS over TLS [RFC7858] or HTTPS [RFC8484], or some other secure mechanism.

If the SRV Resource Record cannot be authenticated, it may be possible for an attacker who can spoof the resource record to perform a denial of service for the receiver by providing wrong or missing authentication metadata. An attacker who can also inject traffic for (S,G)s, would also be able to provide false content in the data stream, so an attacker who can perform both could provide authenticated false content by authenticating with a trust anchor from an attacker-controlled DORMS server.

Clients MAY use other secure methods to explicitly associate an (S,G) with a set of DORMS server hostnames, such as a configured mapping or an alternative trusted lookup service.

8. Acknowledgements

Thanks to Christian Worm Mortensen for some very helpful comments and review.

9. References

9.1. 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>>.

- [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>>.
- [RFC3596] Thomson, S., Huitema, C., Ksinant, V., and M. Souissi, "DNS Extensions to Support IP Version 6", STD 88, RFC 3596, DOI 10.17487/RFC3596, October 2003, <<https://www.rfc-editor.org/info/rfc3596>>.
- [RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <<https://www.rfc-editor.org/info/rfc6991>>.
- [RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <<https://www.rfc-editor.org/info/rfc7950>>.
- [RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <<https://www.rfc-editor.org/info/rfc8040>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8294] Liu, X., Qu, Y., Lindem, A., Hopps, C., and L. Berger, "Common YANG Data Types for the Routing Area", RFC 8294, DOI 10.17487/RFC8294, December 2017, <<https://www.rfc-editor.org/info/rfc8294>>.
- [W3C.REC-cors-20140116] Kesteren, A., "Cross-Origin Resource Sharing", World Wide Web Consortium Recommendation REC-cors-20140116, January 2014, <<http://www.w3.org/TR/2014/REC-cors-20140116>>.

9.2. Informative References

- [I-D.draft-jholland-cb-assisted-cc-01] Holland, J., "Circuit Breaker Assisted Congestion Control (CBACC): Protocol Specification", draft-jholland-cb-assisted-cc-01 (work in progress), April 2017.
- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <<https://www.rfc-editor.org/info/rfc1034>>.

- [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>>.
- [RFC2845] Vixie, P., Gudmundsson, O., Eastlake 3rd, D., and B. Wellington, "Secret Key Transaction Authentication for DNS (TSIG)", RFC 2845, DOI 10.17487/RFC2845, May 2000, <<https://www.rfc-editor.org/info/rfc2845>>.
- [RFC2931] Eastlake 3rd, D., "DNS Request and Transaction Signatures (SIG(0)s)", RFC 2931, DOI 10.17487/RFC2931, September 2000, <<https://www.rfc-editor.org/info/rfc2931>>.
- [RFC3040] Cooper, I., Melve, I., and G. Tomlinson, "Internet Web Replication and Caching Taxonomy", RFC 3040, DOI 10.17487/RFC3040, January 2001, <<https://www.rfc-editor.org/info/rfc3040>>.
- [RFC3376] Cain, B., Deering, S., Kouvelas, I., Fenner, B., and A. Thyagarajan, "Internet Group Management Protocol, Version 3", RFC 3376, DOI 10.17487/RFC3376, October 2002, <<https://www.rfc-editor.org/info/rfc3376>>.
- [RFC3810] Vida, R., Ed. and L. Costa, Ed., "Multicast Listener Discovery Version 2 (MLDv2) for IPv6", RFC 3810, DOI 10.17487/RFC3810, June 2004, <<https://www.rfc-editor.org/info/rfc3810>>.
- [RFC4604] Holbrook, H., Cain, B., and B. Haberman, "Using Internet Group Management Protocol Version 3 (IGMPv3) and Multicast Listener Discovery Protocol Version 2 (MLDv2) for Source-Specific Multicast", RFC 4604, DOI 10.17487/RFC4604, August 2006, <<https://www.rfc-editor.org/info/rfc4604>>.
- [RFC4607] Holbrook, H. and B. Cain, "Source-Specific Multicast for IP", RFC 4607, DOI 10.17487/RFC4607, August 2006, <<https://www.rfc-editor.org/info/rfc4607>>.
- [RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <<https://www.rfc-editor.org/info/rfc6020>>.

- [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, <<https://www.rfc-editor.org/info/rfc6335>>.
- [RFC6415] Hammer-Lahav, E., Ed. and B. Cook, "Web Host Metadata", RFC 6415, DOI 10.17487/RFC6415, October 2011, <<https://www.rfc-editor.org/info/rfc6415>>.
- [RFC7858] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)", RFC 7858, DOI 10.17487/RFC7858, May 2016, <<https://www.rfc-editor.org/info/rfc7858>>.
- [RFC8484] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <<https://www.rfc-editor.org/info/rfc8484>>.
- [whatwg-fetch] Kesteren, A., "WHATWG Fetch Living Standard", August 2019, <<https://fetch.spec.whatwg.org/>>.

Author's Address

Jake Holland
Akamai Technologies, Inc.
150 Broadway
Cambridge, MA 02144
United States of America

Email: jakeholland.net@gmail.com

RTGWG
INTERNET-DRAFT
Intended Status: Informational
Expires: May 7, 2020

Y. Li
J. He
Huawei Technologies
L. Geng
P. Liu
China Mobile
Y. Cui
Tsinghua University
November 4, 2019

Framework of Compute First Networking (CFN)
draft-li-rtgwg-cfn-framework-00

Abstract

Compute First Networking (CFN) leverages both computing and networking status to help determine the optimal edge among multiple edge sites with different geographic locations to serve a specific edge computing request. Requests for the same service can be determined and dispatched to different edges based on service requirements, network and computing resource conditions and other factors to achieve better load balancing and system efficiency. The request needs to be dispatched to the selected edge in real time and the subsequent packets from the same flow should be served by the same edge for flow affinity. This document describes a framework of CFN to achieve the desired features.

Status of this Memo

This Internet-Draft is submitted to IETF in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at
<http://www.ietf.org/lid-abstracts.html>

The list of Internet-Draft Shadow Directories can be accessed at

<http://www.ietf.org/shadow.html>

Copyright and License Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. CFN Framework	4
2.1 CFN Service Overview	5
2.2 Generic Workflow	7
3. Control Plane and Data Plane	7
3.1 Control plane	7
3.2 Data plane	9
4. Summary	12
5. Security Considerations	12
6. IANA Considerations	12
7. Acknowledgements	12
8. References	13
8.1 Normative References	13
8.2 Informative References	13
Authors' Addresses	13

1. Introduction

Compute First Networking (CFN) scenarios and requirements document [CFN-req] shows the usage scenarios that require an edge to be dynamically selected from multiple edge sites with different geographic locations to serve an edge computing request based on computing resource consumption and network status in real time. For instance, edge site in residential area receives low request volume during working hours and high request volume during non-working hours. And the request volume received by the edge site in industrial park is the opposite. Such a pattern causes a big difference of computing load on different edge sites. Traditional static or hashing based service dispatch can not adapt to the unbalanced nature of computing load or rapid change of it on different edge sites. One edge such as the closest one to the client may have been overloaded and at the same time the other edges may still have plenty of computing resources to serve the requests. To efficiently leveraging the computing resources hosted on all edges, service requests should be dispatched and handled dynamically to make the computing and network resources consumed in a balanced way.

CFN assumes there are multiple service equivalent edges to serve a single service. A single edge has limited computing resources and different edges may have different resources available for serving a specific service at a specific time. In concept, multiple edges are interconnected and collaborated with each other to balance the service load in CFN. Computing resource available to serve a request is the top metric to be considered when dispatching a request. At the same time, the quality of the network path to an edge varies over time. CFN is a network based approach so that the request is dispatched to the optimal edge in terms of both computing resources available and network status on the fly.

This document presents a CFN framework which can support service equivalency and dynamics in edge computing to achieve better load balancing with no application dependency.

1.1 Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

CFN: Computing First Networking

2. CFN Framework

Edge computing is expanding from a single edge site to networked and collaborated multiple edge sites to solve the issues of low efficiency and low resource reuse. CFN enables large scale edge interconnection and collaboration, providing optimal service access and load balancing to adapt to service dynamics. Based on the real-time computing capacity available and the network conditions, CFN dynamically schedules computing request to appropriate service node, thus the resource utilization and user experience is improved.

Figure 1 shows the network topology of CFN. CFN node is the basic function entity in CFN network to provide the capability to exchange the information about the computing resource consumption information of service nodes attached to it and/or provide the CFN service access to the clients. Edge site (edge for short) is normally the site where the edge computing is hosted. CFN node can be a network virtual function (NFV) co-located with service node in a server. CFN node's function can also be provided by physical equipment like access router in access ring or metro network.

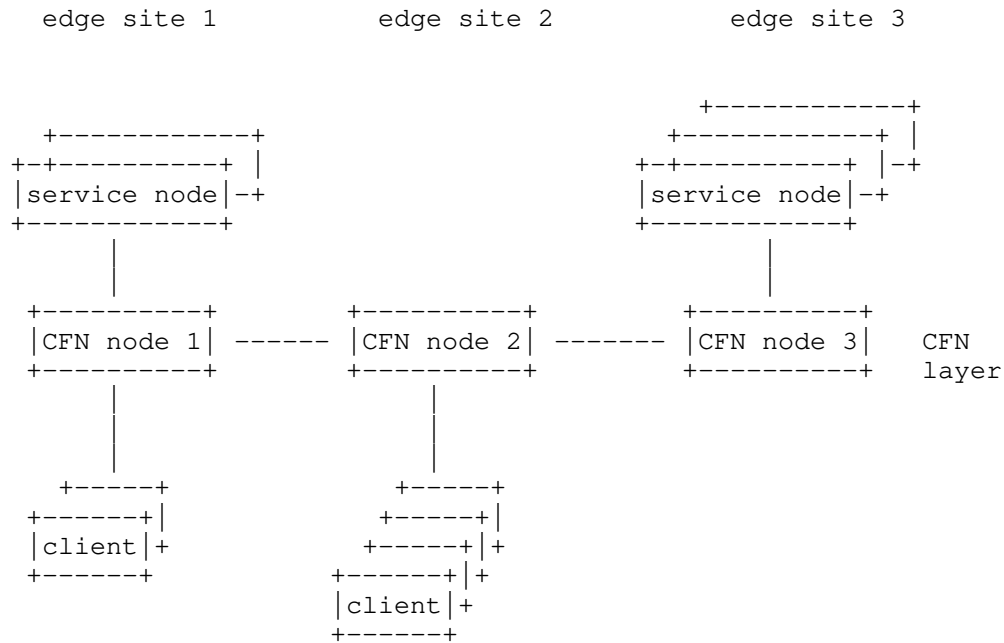


Figure 1. CFN Network Topology

2.1 CFN Service Overview

CFN uses Service ID (SID) to identify a particular service provided by service nodes on multiple edges. The end devices always use SID to initiate an access to a service. SID in current system is an anycast address. Request to a single SID can potentially be served by different edge sites. The end device does not know in advance which edge to serve the request. The procedures to make such determination is called the service dispatch. During service dispatch, the most appropriate edge site (i.e. CFN egress) is selected and it is the edge to which the service node that handles this specific request is attached to. A binding IP (BIP) address to the requested SID is known by CFN egress. BIP is a unicast IP address accessible to a particular service node providing service SID.

As shown in figure 2, service with SID 2 can be served by either CFN node 2 with binding IP BIP22 or CFN node 3 with BIP32. When the service request from the end device to SID2 reaches the ingress CFN (which is CFN node 1 in this case), the ingress CFN node should determine on the fly which egress CFN this request should be sent to. Then, the de facto service node is determined, and all the subsequent

data packets from the same flow to access this service should always be sent to the binding IP of the selected service node.

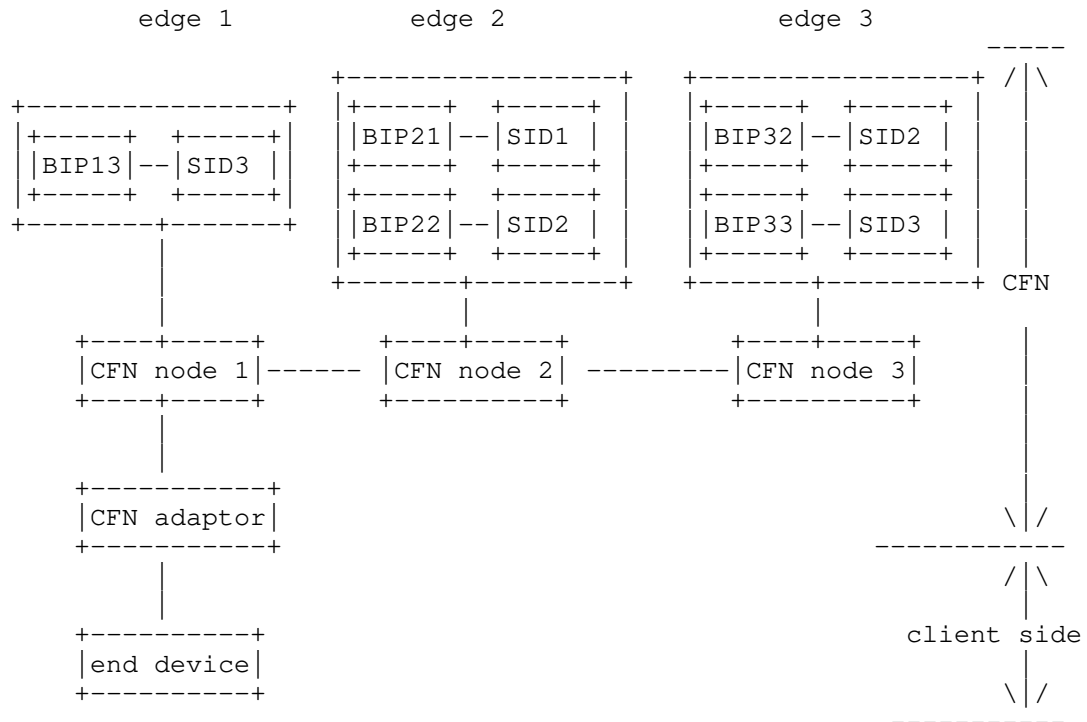


Figure 2. CFN System Overview

CFN adaptor shown in figure 2 is an entity to help the end device working with CFN in a way of keeping the binding information, identifying the initial request packet, and so on. It can be implemented as a part of CFN node (internal mode) or on a separate equipment (external mode). Figure 2 shows an external mode of CFN adaptor which can be deployed at the client side, on a virtual gateway connecting multi user equipments (UEs), as a user Plane Function (UPF) in the mobile network, or on Broadband Remote Access Server (BRAS) in the fixed network. The reason to have such an external mode is that CFN adaptor can be put closer to the clients, and then CFN node is put at some aggregated point with multiple CFN adaptors attached to it. Compared to the internal mode, external CFN adaptor keeps less binding information of the clients. It results in

less memory requirements on CFN node. CFN adaptor has no control plane.

2.2 Generic Workflow

The following procedures describe how CFN works in general.

- 1) CFN adaptor identifies a new service request from the end device, possibly by the special anycast address range for a SID.
- 2) CFN adaptor sends the request to its attaching CFN node which is CFN ingress.
- 3) CFN ingress determines the most appropriate CFN egress based on the computing resource consumption of the service nodes, the network status to the egress nodes and other information. CFN ingress forwards the request to the selected CFN egress. CFN ingress can select itself to serve the request. In this case, it is both ingress and egress in concept.
- 4) CFN egress receives the request from the CFN ingress and explicitly uses the binding IP BIP as destination address to access the required service.
- 5) CFN adaptor of ingress keeps the binding information on (SID, CFN egress) for the flow.
- 6) CFN ingress sends the subsequent packets for the same service from the same flow to the bound CFN egress to ensure the flow affinity.
- 7) CFN nodes distribute the service nodes status like available computing resources for specific services to each other on a regular base.

3. Control Plane and Data Plane

3.1 Control plane

CFN node needs to notify each other about service IDs (SIDs) attaching to it and the computing load information available corresponding to each service ID. This is used for service discovery and dispatch when a request to access a SID is received. Such information can be carried in current BGP [RFC4760] /IGP routing protocol extension. The network cost to a CFN node can be distributed in the same way. A sample service status information to be stored on

a CFN edge is shown in figure 2.

Destination	Computing Load	Network Cost	Next Hop
SID 1	3	5	CFN Egress node 1

Figure 2. Example of service status information in CFN

Computing load can be calculated from different weighted dimensions, e.g. CPU used, number of session being served, query per second, computation delay and so on. Such information needs to be refreshed regularly. In order to avoid fluctuation, it is distributed only when the metrics variation exceeds a threshold or the updating timer is expired. At the same time, the most appropriate egress node selected by the CFN ingress does not necessarily mean the one with the lowest load. Request can be sent to one selected from those egresses with relatively low computing load to avoid fluctuation.

Since SID is an anycast address, CFN ingress determines which CFN egress to forward the request to a specific SID to based on a combination of computing load and network cost.

Figure 3 shows how CFN control plane works in general. It depicts that CFN node 3 distributes computing information for service SID2. CFN node 2 should distribute service SID 2 information in the similar way as shown in figure 3. Definition and operations to extend control plane routing protocol to support CFN information distribution, and schemes/criteria to select CFN egress with anycast address from those information are to be added.

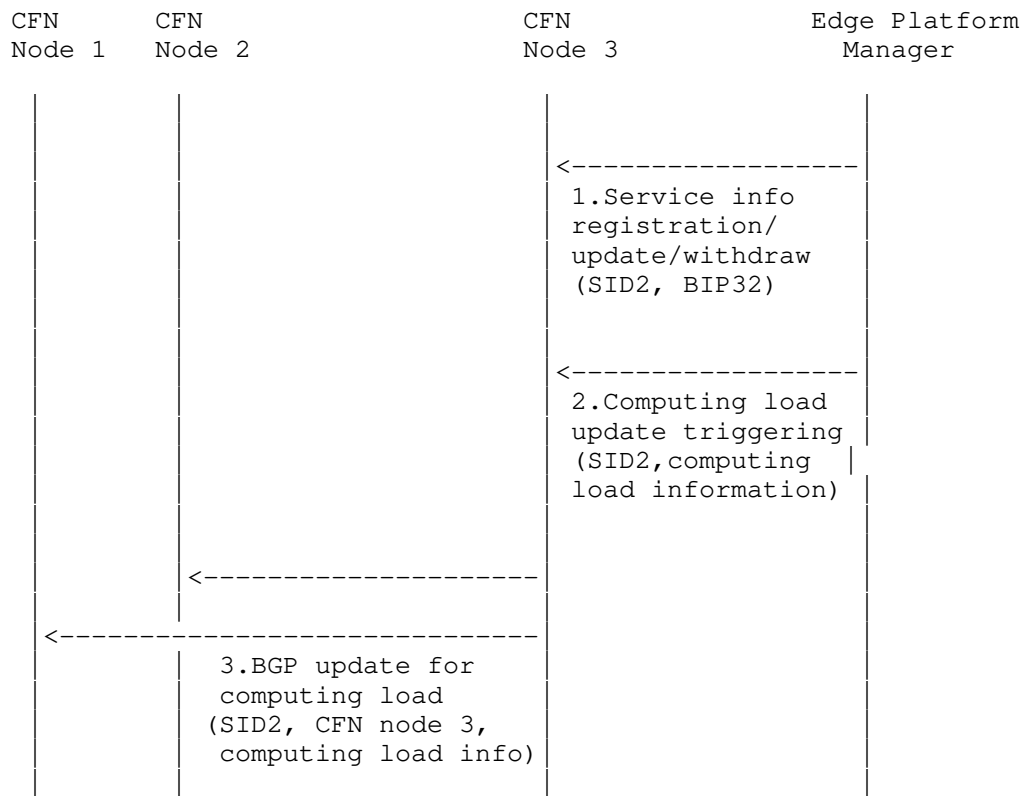


Figure 3. CFN control plane

3.2 Data plane

The traditional anycast is normally used for single request single response style communication as different requests may be sent to different places when the network status changes. CFN used in edge computing may require multiple request multiple response style communication between the end device and the service node. Therefore the data plane must maintain flow affinity to ensure that the requests from the same flow are always processed by the same edge and that edge is determined at the time when the first anycast request is received by CFN ingress. The service access to the same SID from different end hosts attaching to the same CFN ingress may be dispatched to different CFN egresses. We call such a feature dynamic anycast or Dynicast in this document.

Dynacast puts some requirements on the data plane. The flow affinity table needs to be maintained by CFN ingress. On the other hand, large number of end hosts may attach to a CFN node. Therefore CFN ingress may require large memory space, such as tens of thousands of entries, to maintain such a big table of (flow, service ID, egress CFN). It is preferable to place such a binding table on an external CFN adaptor as CFN adaptor only needs to maintain a much smaller table, usually less than a hundred.

Figure 4 shows how CFN data plane works in general.

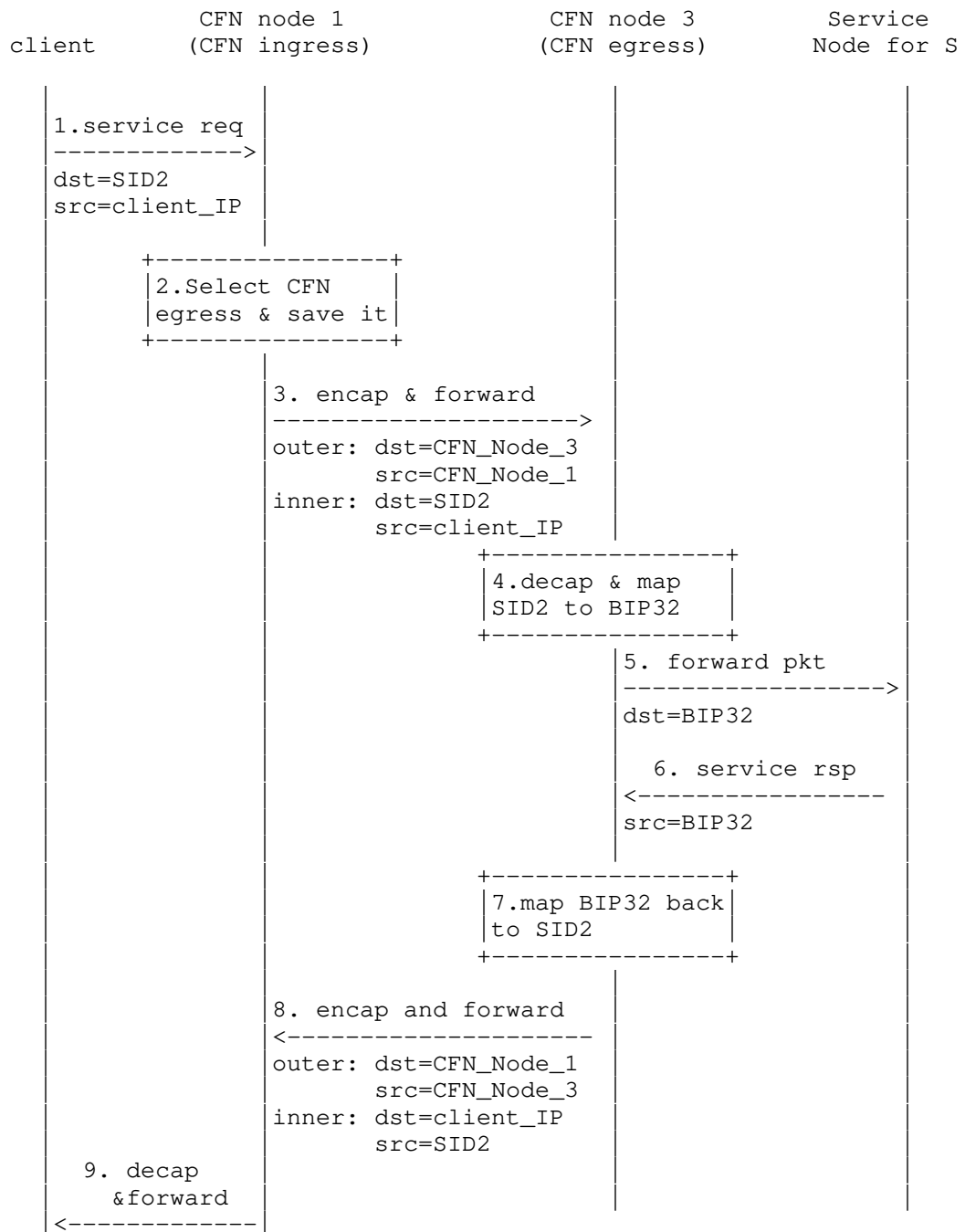


Figure 4. CFN data plane for the first request of a flow

The data plane supports the following functions.

- CFN ingress forwards the first service access request packet of a flow to the selected CFN egress by encapsulation, source routing or segment routing. Figure 4 shows the example of forwarding by encapsulation.
- CFN ingress can inform the external CFN adaptor (if there is) about the binding information on (flow, service ID, egress CFN).
- CFN adaptor (internal or external to CFN ingress) maintains the binding information table for all end hosts attaching to it and forwards the subsequent packets based on the binding information if any.

4. Summary

This draft introduces a CFN framework that enables the service request to be sent to an optimal edge to improve the overall system load balancing. It can dynamically adapt to the computing resources consumption and network status on edges and avoid overloading a single load. CFN is a network based solution that supports a large number of edges and is independent of the applications or services hosted on the edge.

This present document is a strawman for defining CFN framework. A routing protocol (BGP [RFC4760]/IGP based) extension to distribute computing resource information and a late binding based dynamic anycast are to be defined on control plane and data plane respectively.

5. Security Considerations

The computing resource information changes over time very fast with the creation and termination of service instance handlers. When such information is carried in routing protocol, too many updates can make the network fluctuate. Section 3.1 gives a brief idea on avoiding sending too much updates.

6. IANA Considerations

No IANA action is required so far.

7. Acknowledgements

8. References

8.1 Normative References

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

8.2 Informative References

- [RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, January 2007.
- [CFN-req] Geng, L., et al, "Compute First Networking (CFN) Scenarios and Requirements", draft-geng-rtgwg-CFN-req-00, November 2019.

Authors' Addresses

Yizhou Li
Huawei Technologies

Email: liyizhou@huawei.com

Jeffrey He
Huawei Technologies

Email: jeffrey.he@huawei.com

Liang Geng
China Mobile
Email: gengliang@chinamobile.com

Peng Liu
China Mobile
Email: liupengyjy@chinamobile.com

Yong Cui
Tsinghua University

Email: cuiyong@tsinghua.edu.cn

HIP
Internet-Draft
Intended status: Standards Track
Expires: 10 September 2020

R. Moskowitz
HTT Consulting
S. Card
A. Wiethuechter
AX Enterprize
9 March 2020

Hierarchical HIT Registries
draft-moskowitz-hip-hhit-registries-02

Abstract

This document describes using the registration protocol and registries to support hierarchical HITs (HHITs). New and existing HIP parameters are used to communicate Registry Policies and data about the HHIT device and the Registries. Further Registries are expected to provide RVS services for registered HHIT devices.

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 <https://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 10 September 2020.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

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

as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terms and Definitions	3
2.1. Requirements Terminology	3
2.2. Definitions	3
3. Problem Space	3
3.1. Desire for administrative control of HHITs	4
3.2. Desire for administrative control by RVS providers	4
3.3. Defense against fraudulent HITs	4
4. HHIT Registry services to support hierarchical HITs	4
4.1. Hierarchical HIT Registration using X.509 Certificates	5
4.2. Hierarchical HIT Registration using a PSK	5
4.3. HIP Parameters	5
4.3.1. CERT Parameter	6
4.3.2. Hierarchical HIT Registration Type	6
4.3.3. Hierarchical HIT Registration Failure Type	6
4.3.4. CLIENT_INFO	6
4.4. Registration failure behavior	7
4.4.1. Example of a simple HDA policy	7
4.5. Example of a simple HDA policy	7
4.6. HHIT DNS Retrieval Service	8
5. Using hierarchical HITs	8
5.1. Contacting a HIP client	8
5.2. Defense against fraudulent HITs	9
6. IANA Considerations	9
7. RAA Management Organization Considerations	9
8. Security Considerations	9
8.1. Privacy Concerns	10
9. Acknowledgments	11
10. References	11
10.1. Normative References	11
10.2. Informative References	11
Appendix A. Calculating Collision Probabilities	12
Authors' Addresses	12

1. Introduction

This document expands on Hierarchical HITs [I-D.moskowitz-hip-hierarchical-hit], defining HIP registration protocol enhancements, the Registry services to support hierarchical HITs (HHITs), and given a HHIT, how to get additional information about the device.

Registries will tend to be organized regionally and by nature of their clients. For example, an RAA may be US centric and only have HDAs that are US-based.

Registries will need to work in a federation. Devices that are clients of one HDA/RAA will be needing information and connectivity to devices that are clients of other HDA/RAA. The policies for establishing such federations are outside the scope of this document.

Access to information at a Registry about a device may require authorization. The nature and process of such an authorization is outside the scope of this document.

2. Terms and Definitions

2.1. Requirements Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2.2. Definitions

CSR (Certificate Signing Request): Request to a Certificate Authority to create an X.509 certificate with the provided information.

HDA (Hierarchical HIT Domain Authority): The 14 bit field identifying the HIT Domain Authority under a RAA.

HID (Hierarchy ID): The 32 bit field providing the HIT Hierarchy ID.

RAA (Registered Assigning Authority): The 18 bit field identifying the Hierarchical HIT Assigning Authority.

RVS (Rendezvous Server): The HIP Rendezvous Server for enabling mobility, as defined in [RFC8004].

3. Problem Space

3.1. Desire for administrative control of HHITs

For HHITs to be effectively used, the HHIT Domain Authorities (HDAs) need to provide information on the HHIT devices. Minimally this would be the corresponding HI, information on the device owner (only to authorized requesters), and where in the network the device has last reported from.

The HHIT space creates a type of a business labeling for the HDAs. "These are my customers."

3.2. Desire for administrative control by RVS providers

An RVS [RFC8004] provider may only be willing to provide discovery (RVS) services to HIP devices it knows and trusts. A flat HIT space does not provide any intrinsic functionality to support this. A HHIT space can be mapped to the RVS provider. DNS can effectively be used to provide the HHIT to IP mapping without Distributed Hash Table (DHT) [RFC6537].

3.3. Defense against fraudulent HITs

How can a host protect against a fraudulent HIT? That is, a second pre-image attack on the HI hash that produces the HIT. A strong defense would require every HIT/HI registered and openly verifiable. With HHITs, the HDAs can provide the HI and proof of registration (e.g. X.509 certificate including HHIT).

This would best be done as either part of the R1 and I2 validation, or anytime a HHIT is presented.

4. HHIT Registry services to support hierarchical HITs

Hierarchical HIT registration SHOULD be performed using the HIP Registration Extension [RFC8003]. The client either uses an X.509 certificate [RFC8002], or use a PSK, as defined in Appendix A of HIP-DEX [I-D.ietf-hip-dex], to validate the registration.

The Registration should include additional client information. This information may be contained within the X.509 certificate (CERT parameter) and/or is carried in the CLIENT_INFO parameter, see Section 4.3.4. The Registrar can include its requirements in the R1 packet, and the client provide its information in the I2 packet. This parameter may be encrypted within the ENCRYPTED parameter. If the CLIENT_INFO contains Personal Identifying Information (PII), then it MUST be placed into the ENCRYPTED parameter.

The content and internal format of the CLIENT_INFO parameter is set by the HDA's policy and is outside the scope of this document. Examples of client information can be by phone number, IMEI, and ICCID.

4.1. Hierarchical HIT Registration using X.509 Certificates

This requires the HIP client to possess a client certificate trusted by the HDA/Registrar. Registration will either succeed or fail.

Certificate registration can be a "chicken and egg" problem: where did the device get its certificate? Thus this is more likely used in a re-registration situation with updated information.

4.2. Hierarchical HIT Registration using a PSK

This requires the HIP client and the HDA/Registrar to share a PSK. The PSK is carried in the ENCRYPTED_KEY parameter [I-D.ietf-hip-dex]. The PSK may already exist prior to starting the registration and just be used within the registration. A PSK out-of-band exchange may be triggered by performing the registration without any authentication.

If no client authentication is included in the I2 packet, the registration fails with "No Authentication provided". If the I2 packet included the proper HDA required client information, the HDA can use it to set up a side channel for an out-of-band delivery of a PSK. An example of this would be to send an SMS message with the PSK. Once the client possesses the PSK, it can rerun the registration at which point the HI and HIT duplicate checks are performed.

The I2 packet may contain a CERT parameter containing a CSR, and the R2 would return the X.509 certificate for later use.

4.3. HIP Parameters

The HIP parameters carry information that is necessary for establishing and maintaining a HIP association. For example, the device's public keys as well as the signaling for negotiating ciphers and payload handling are encapsulated in HIP parameters. Additional information, meaningful for end hosts or middleboxes, may also be included in HIP parameters. The specification of the HIP parameters and their mapping to HIP packets and packet types is flexible to allow HIP extensions to define new parameters and new protocol behavior.

4.3.1. CERT Parameter

The CERT parameter, [RFC8002], is a container for certain types of digital certificates.

A new CERT Type, CSR, is defined here. When CERT Type is CSR, CERT ID is Zero. There is only ONE CSR in a CERT Parameter.

CERT format	Type number	RFC
-----	-----	----
PKCS#10 - CSR	9	2986

4.3.2. Hierarchical HIT Registration Type

The Registration Type used in the REG_REQUEST is:

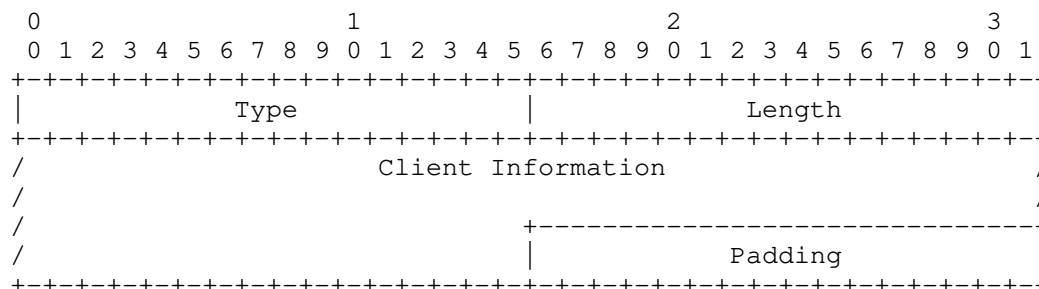
Number	Registration Type
-----	-----
2	HHIT Registration

4.3.3. Hierarchical HIT Registration Failure Type

The Registration may fail. In fact, with PSK, this may be the response to expect an SMS message with the PSK to use in a second registration request. Failure Types used in the REG_FAIL are:

Failure Type	Reason
-----	-----
[TBD-IANA]	Hierarchical HIT Already Registered
[TBD-IANA]	HI Already Registered
[TBD-IANA]	Previously Registered HI with different device information
[TBD-IANA]	No Authentication provided
[TBD-IANA]	Invalid Authentication
[TBD-IANA]	Invalid Authentication, new PSK sent via SMS

4.3.4. CLIENT_INFO



Type	[TBD-IANA]
Length	length in octets, excluding Type, Length, and Padding
Client Information	The information required by the HDA in the format required by the HDA.

This parameter contains client information to include in the HIT registration. The specific content and format is set by the HDA.

4.4. Registration failure behavior

If the failure type is "Hierarchical HIT Already Registered", the client's HI is hashing to an existing HIT and must generate a new HI and hierarchical HIT and re-register. If the failure is "HI Already Registered", the client should assume it is registered. If the failure is "Previously Registered HI with different device information", either the client managed to generate a duplicate HI, possibly indicating a weak key generation algorithm, or the client was previously registered on a different device. Resolving this conflict will be left to the HDA's policy.

4.4.1. Example of a simple HDA policy

A simple HDA policy would be to require the device to generate a new HI and thus HHIT and try registration again. The HDA policy may also provide a URL for "Previous Registration Resolution". This contact is primarily to assist a device that was registered, but had some local failure resulting in a new registration attempt.

4.5. Example of a simple HDA policy

A simple HDA policy would be to require the device to generate a new HI and thus HHIT and try registration again. The HDA policy may also provide a URL for "Previous Registration Resolution". This contact is primarily to assist a device that was registered, but had some local failure resulting in a new registration attempt.

4.6. HHIT DNS Retrieval Service

A Registry SHOULD provide DNS retrieval services for the HIP RR [RFC8005] for HHITs as described in Hierarchical HITs [I-D.moskowitz-hip-hierarchical-hit].

This requires a Registry to act as a DNS zone Name Server to provide minimally the HI for the HHIT in the DNS query. Registry policy will determine if the response can be cached within DNS. If the Registry also provides the HHIT and/or the RVS for the HHIT, this may result in a different DNS caching policy by the Registry.

5. Using hierarchical HITs

All HIP clients with hierarchical HITs maintain an RVS connection with their HDA's RVS server(s). How the HDA scales this service up to a potential population in the millions is out of scope of this document. Lifetime management of these connections is also out of scope.

One approach an HDA can use to address the scaling challenge is to add an internal level of hierarchy to assign a set number of devices per RVS server.

Peering agreements between HDAs would allow for geographically close RVS to a device. This may reduce the latency for use of a device's current RVS. This is a subject of another document.

5.1. Contacting a HIP client

A service Initiator uses some service to discover the HIT of the service Responder. The Initiator uses the hierarchical information in the HIT to find the Responder's RVS. A trusted RVS discover method could use the DNS PTR to RVS as shown in Hierarchical HITs [I-D.moskowitz-hip-hierarchical-hit]. An I1 is sent to that RVS which forwards it to the Responder.

The potential Responder uses the HIT in the I1 to query the Initiator's RVS about the Initiator. The nature of information, and method of communication are determined by the Initiator's HDA and the Responder's (and or HDA's) relationship with it. Based on the Responder's local policy, this information will be used to determine if the contact is to be accepted. If accepted, the Responder may proceed sending an R1 to the Initiator. It may alternatively initiate some non-HIP process.

It should be noted that this R1 may contain a REG_INFO list for the

Initiator to validate that the Responder does offer the desired service.

5.2. Defense against fraudulent HITs

Both the Initiator and Responder MAY validate a peer host as a defense against a second pre-image attack on the HHIT. This may occur via a CERT [RFC8002] in R1 or I2. It may be through a back end process associated with the R1 or I2 validation to look up the HHIT and retrieve the registered HI.

6. IANA Considerations

IANA will need to make the following changes to the "Host Identity Protocol (HIP) Parameters" registries:

CERT Type: This document defines the new CERT Type for the CERT parameter "PKCS#10 - CSR" (see Section 4.3.1).

Reg Type: This document defines the new Registration Type for the REG_REQUEST parameter "HIT Registration" (see Section 4.3.2).

Reg Fail: This document defines the new Failure Types for the REG_FAIL parameter (see Section 4.3.3).

CLIENT_INFO: This document defines the new CLIENT_INFO parameter (see Section 4.3.4). The parameter value will be assigned by IANA.

7. RAA Management Organization Considerations

Introducing the RAA management organization may be the largest hurdle for hierarchical HITs. Thus it would be best if this were adopted by an organization already in the business of allocating numbers within either the Internet or the Mobile, cellular, infrastructure.

One consideration would be to reserve the first N RAA values to map to the existing DNS TLDs. For example, these TLDs can be organized in an ascending order and numbered accordingly. Thus the 2 character TLDs will be a lower number than the 3 character TLDs. After that, it could be a first come, first numbered assignment process.

8. Security Considerations

There are potential risks with the hierarchical HIT, the Registry service, and the discovery of potential peer hosts using its hierarchical HIT.

A 64 bit hash space presents a real risk of second pre-image attacks. The HHIT Registry services effectively block attempts to "take over" a HHIT. It does not stop a rogue attempting to impersonate a known HHIT. This attack can be mitigated by the Responder using DNS to find the HI for the HHIT or the RVS for the HHIT that then provides the registered HI.

The two risks with hierarchical HITs are the use of an invalid HID and forced HIT collisions. The use of the "hhit.arpa." DNS zone is a strong protection against invalid HIDs. Querying an HDA's RVS for a HIT under the HDA protects against talking to unregistered clients. The Registry service has direct protection against forced or accidental HIT hash collisions.

By using the HIP Registration Extension, the Registry service is protected from direct attacks. This service does rely on either the integrity of a PKI service or an out-of-band PSK delivery process. Thus the risk to the Registry service is highly related to the trust in these authentication setup services. Further, the duplicate HI resolution process may require human interaction with related social engineering risks.

Finally the peer host discovery process relies on trusting the finding the proper HDA for the host and its forwarding the I1 to the proper Responder. A rogue RVS, impersonating the RVS for the HIT, could redirect the I1 to a client that has forced a collision with the HIT and the Initiator would be none the wiser. The only defense against this is if the Initiator has some other source for the Responder HI and validate the HI in the R1.

8.1. Privacy Concerns

Mobile-privacy-attack [I-D.moskowitz-mobile-privacy-attack] details how Eve can follow a communication between two mobile peers using the session Identifiers and deep knowledge about those Identifiers gained by hacking servers that log PII related to the Identifiers.

Hierarchical HITs not only does not mitigate this attack, it can actually aggravate it by supplying the HDA where the HHIT is registered.

A HIP Privacy Enhanced Base Exchange, to be defined in a separate draft, along with a Privacy Enhanced ESP tunnel, can be used to hide all the HIP and ESP Identifiers from Eve.

9. Acknowledgments

Sue Hares of Huawei contributed to the clarity in this document.

10. References

10.1. 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>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

10.2. Informative References

- [I-D.ietf-hip-dex] Moskowitz, R., Hummen, R., and M. Komu, "HIP Diet EXchange (DEX)", Work in Progress, Internet-Draft, draft-ietf-hip-dex-13, 14 February 2020, <<https://tools.ietf.org/html/draft-ietf-hip-dex-13>>.
- [I-D.moskowitz-hip-hierarchical-hit] Moskowitz, R., Card, S., and A. Wiethuechter, "Hierarchical HITs for HIPv2", Work in Progress, Internet-Draft, draft-moskowitz-hip-hierarchical-hit-04, 3 March 2020, <<https://tools.ietf.org/html/draft-moskowitz-hip-hierarchical-hit-04>>.
- [I-D.moskowitz-mobile-privacy-attack] Moskowitz, R., "An Attack on Privacy in Mobile Devices", Work in Progress, Internet-Draft, draft-moskowitz-mobile-privacy-attack-01, 13 November 2017, <<https://tools.ietf.org/html/draft-moskowitz-mobile-privacy-attack-01>>.
- [RFC6537] Ahrenholz, J., "Host Identity Protocol Distributed Hash Table Interface", RFC 6537, DOI 10.17487/RFC6537, February 2012, <<https://www.rfc-editor.org/info/rfc6537>>.
- [RFC8002] Heer, T. and S. Varjonen, "Host Identity Protocol Certificates", RFC 8002, DOI 10.17487/RFC8002, October 2016, <<https://www.rfc-editor.org/info/rfc8002>>.

- [RFC8003] Laganier, J. and L. Eggert, "Host Identity Protocol (HIP) Registration Extension", RFC 8003, DOI 10.17487/RFC8003, October 2016, <<https://www.rfc-editor.org/info/rfc8003>>.
- [RFC8004] Laganier, J. and L. Eggert, "Host Identity Protocol (HIP) Rendezvous Extension", RFC 8004, DOI 10.17487/RFC8004, October 2016, <<https://www.rfc-editor.org/info/rfc8004>>.
- [RFC8005] Laganier, J., "Host Identity Protocol (HIP) Domain Name System (DNS) Extension", RFC 8005, DOI 10.17487/RFC8005, October 2016, <<https://www.rfc-editor.org/info/rfc8005>>.

Appendix A. Calculating Collision Probabilities

The accepted formula for calculating the probability of a collision is:

$$p = 1 - e^{\{-k^2/(2n)\}}$$

P Collision Probability
n Total possible population
k Actual population

Authors' Addresses

Robert Moskowitz
HTT Consulting
Oak Park, MI 48237
United States of America

Email: rgm@labs.htt-consult.com

Stuart W. Card
AX Enterprize
4947 Commercial Drive
Yorkville, NY 13495
United States of America

Email: stu.card@axenterprize.com

Adam Wiethuechter
AX Enterprize
4947 Commercial Drive
Yorkville, NY 13495

United States of America

Email: adam.wiethuechter@axenterprize.com

HIP
Internet-Draft
Updates: 7401 (if approved)
Intended status: Standards Track
Expires: 13 November 2020

R. Moskowitz
HTT Consulting
S. Card
A. Wiethuechter
AX Enterprize
12 May 2020

Hierarchical HITs for HIPv2
draft-moskowitz-hip-hierarchical-hit-05

Abstract

This document describes using a hierarchical HIT to facilitate large deployments of managed devices. Hierarchical HITs differ from HIPv2 flat HITs by only using 64 bits for mapping the Host Identity, freeing 32 bits to bind in a hierarchy of Registering Entities that provide services to the consumers of hierarchical HITs.

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 <https://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 13 November 2020.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terms and Definitions	3
2.1. Requirements Terminology	3
2.2. Definitions	3
3. Problem Space	3
3.1. Meeting the future of Mobile Devices in a public space .	4
3.2. Semi-permanency of Identities	4
3.3. Managing a large flat address space	4
3.4. Defense against fraudulent HITs	5
4. The Hierarchical Host Identity Tag (HHIT)	5
4.1. HHIT prefix	5
4.2. HHIT Suite IDs	6
4.3. The Hierarchy ID (HID)	6
4.3.1. The Registered Assigning Authority (RAA)	6
4.3.2. The Hierarchical HIT Domain Authority (HDA)	6
4.3.3. Example of the HID DNS	7
4.3.4. HHIT DNS Retrieval	7
4.3.5. Changes to ORCHIDv2 to support Hierarchical HITs . .	7
4.3.6. Collision risks with Hierarchical HITs	8
5. IANA Considerations	8
6. Security Considerations	8
7. Acknowledgments	9
8. References	9
8.1. Normative References	9
8.2. Informative References	9
Appendix A. Calculating Collision Probabilities	11
Authors' Addresses	11

1. Introduction

This document expands on HIPv2 [RFC7401] to describe the structure of a hierarchical HIT (HHIT). Some of the challenges for large scale deployment addressed by HHITs are presented. The basics for the hierarchical HIT registries are defined here.

Including hierarchy information within the HIT is not a new concept. This was part of the original HIPv1 Architecture [draft.moskowitz-hip-arch-02]. It was dropped from the HIPv1 work for lack of a use case and concerns over the smaller HI mapping space. It was later brought up in the HIP Research Group (HIP-RG) in [draft.zhang-hip-hierarchical-parameter-00], but this never gained consensus.

Hierarchical HITs now have a solid use case with Public, mobile devices (e.g. Unmanned Aircraft). The math to evaluate the statistical collision risk is available, Appendix A. And finally, HHIT Registries [hhit-registries] provide a way to manage the hierarchy.

2. Terms and Definitions

2.1. Requirements Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2.2. Definitions

HDA (Hierarchical HIT Domain Authority):

The 16 bit field identifying the HIT Domain Authority under an RAA.

HID (Hierarchy ID):

The 32 bit field providing the HIT Hierarchy ID.

RAA (Registered Assigning Authority):

The 16 bit field identifying the Hierarchical HIT Assigning Authority.

RVS (Rendezvous Server):

The HIP Rendezvous Server for enabling mobility, as defined in [RFC8004].

3. Problem Space

3.1. Meeting the future of Mobile Devices in a public space

Public safety may impose a "right to know" what devices are in a public space. Public space use may only be permitted to devices that meet an exacting "who are you" query. This implies a device identity that can be quickly validated by public safety personnel and even the general public in many situations.

Many proposals for mobile device identities are nothing more than a string of bits. These may provide information about the device but provide no assurance that the identity associated with a device really belongs to a particular device; they are highly susceptible to fraudulent use. Further they may impose a slow, complex method to discover the device owner to those with appropriate authorization.

The Host Identity Tag (HIT) from the Host Identity Protocol (HIP) provides a self-asserting Identity through a public key signing operation using the Host Identity's (HI) private key.

Although the HIT provides a "trust me, I am me" claim, it does not provide an assertion as to why the claim should be trusted and any additional side information about the device. The later could be distributed directly from the device in a secure manner, but again there is no 3rd-party assertion of such a claim.

3.2. Semi-permanency of Identities

A device Identity has some degree of permanency. A device creates its identity and registers it to some 3rd-party that will assert a level of trust for that identity. A device may have multiple identities to use in different contexts, and it may deprecate an identity for any number of reasons. The asserting 3rd-party may withdraw its assertion of an identity for any number of reasons. An identity system needs to facilitate all of this.

3.3. Managing a large flat address space

For HITs to be successfully used by a large population of mobile devices, they must support an Identity per device; potentially 10 billion Identities. Perhaps a Distributed Hash Table [RFC6537] can scale this large. There are still the operational challenges in establishing such a world-wide DHT implementation and how RVS [RFC8004] works with such a large population. There is also the challenge of how to turn this into a viable business. How can different controlling jurisdictions operate in such an environment?

Even though the probability of collisions with 7B HITs (one HIT per person) in a 96 bit flat address space is $3.9E-10$, it is still real. How are collisions managed? It is also possible that weak key uniqueness, as has been shown in deployed TLS certificates [WeakKeys], results in a much greater probability of collisions. Thus resolution of collisions needs to be a feature in a global namespace.

3.4. Defense against fraudulent HITs

How can a host protect against a fraudulent HIT? That is, a second pre-image attack on the HI hash that produces the HIT. A strong defense would require every HIT/HI registered and openly verifiable. This would best be done as part of the R1 and I2 validation. Or any other message that is signed by the HI private key.

4. The Hierarchical Host Identity Tag (HHIT)

The Hierarchical HIT (HHIT) is a small but important enhancement over the flat HIT space. By adding two levels of hierarchical administration control, the HHIT provides for device registration/ownership, thereby enhancing the trust framework for HITs.

HHITs represent the HI in only a 64 bit hash and uses the other 32 bits to create a hierarchical administration organization for HIT domains. Hierarchical HITs are "Using cSHAKE in ORCHIDs" [new-orchid]. The input values for the Encoding rules are in Section 4.3.5.

A HHIT is built from the following fields:

- * 28 bit IANA prefix
- * 4 bit HIT Suite ID
- * 32 bit Hierarchy ID (HID)
- * 64 bit ORCHID hash

4.1. HHIT prefix

A unique 28 bit prefix for HHITs is recommended. It clearly separates the flat-space HIT processing from HHIT processing per Section 4 of "Using cSHAKE in ORCHIDs" [new-orchid].

4.2. HHIT Suite IDs

The HIT Suite IDs specifies the HI and hash algorithms. Any HIT Suite ID can be used for HHITs, provided that the prefix for HHITs is different from flat space HITs. Without a unique prefix, Section 4.1, additional HIT Suite IDs would be needed for HHITs. This would risk exhausting the limited Suite ID space of only 15 IDs.

4.3. The Hierarchy ID (HID)

The Hierarchy ID (HID) provides the structure to organize HITs into administrative domains. HIDs are further divided into 2 fields:

- * 16 bit Registered Assigning Authority (RAA)
- * 16 bit Hierarchical HIT Domain Authority (HDA)

4.3.1. The Registered Assigning Authority (RAA)

An RAA is a business or organization that manages a registry of HDAs. For example, the Federal Aviation Authority (FAA) could be an RAA.

The RAA is a 16 bit field (65,536 RAAs) assigned by a numbers management organization, perhaps ICANN's IANA service. An RAA must provide a set of services to allocate HDAs to organizations. It must have a public policy on what is necessary to obtain an HDA. The RAA need not maintain any HIP related services. It must maintain a DNS zone minimally for discovering HID RVS servers.

This DNS zone may be a PTR for its RAA. It may be a zone in a HHIT specific DNS zone. Assume that the RAA is 100. The PTR record could be constructed:

```
100.hhit.arpa    IN PTR      raa.bar.com.
```

4.3.2. The Hierarchical HIT Domain Authority (HDA)

An HDA may be an ISP or any third party that takes on the business to provide RVS and other needed services for HIP enabled devices.

The HDA is an 16 bit field (65,536 HDAs per RAA) assigned by an RAA. An HDA should maintain a set of RVS servers that its client HIP-enabled customers use. How this is done and scales to the potentially millions of customers is outside the scope of this document. This service should be discoverable through the DNS zone maintained by the HDA's RAA.

An RAA may assign a block of values to an individual organization. This is completely up to the individual RAA's published policy for delegation.

4.3.3. Example of the HID DNS

HID related services should be discoverable via DNS. For example the RVS for a HID could be found via the following. Assume that the RAA is 100 and the HDA is 50. The PTR record is constructed as:

```
50.100.hhit.arpa    IN PTR      rvs.foo.com.
```

The RAA is running its zone, 100.hhit.arpa under the hhit.arpa zone.

4.3.4. HHIT DNS Retrieval

The HDA SHOULD provide DNS retrieval per [RFC8005]. Assume that the Host_ID suite of EdDSA25519 (5), RAA of 10 and the HDA of 20 and the HHIT example is:

```
2001:5:a:14:a3ad:1952:ad0:a69e
```

The HHIT FQDN is:

```
2001:0005:a:14:a3ad:1952:0ad0:a69e.20.10.hhit.arpa.
```

The NS record for the HDA zone is constructed as:

```
20.10.hhit.arpa    IN NS      registry.foo.com.
```

registry.foo.com returns a HIP RR with the HHIT and matching HI. The HDA sets its policy on TTL for caching the HIP RR. Optionally, the HDA may include RVS information. Including RVS in the HIP RR may impact the TTL for the response.

4.3.5. Changes to ORCHIDv2 to support Hierarchical HITs

A new format for ORCHIDs to support Hierarchical HITs is defined in "Using cSHAKE in ORCHIDs" [new-orchid]. For this use the following values apply:

```

Prefix      := HHIT Prefix
              Note: per section 4.1, this should be different
                  than the Prefix for RFC 7401
OGA ID      := 4-bit Orchid Generation Algorithm identifier
              The HHIT Suite ID
Context ID  := 0x00B5 A69C 795D F5D5 F008 7F56 843F 2C40
Info (n)    := 32 bit HID (Hierarchy ID)
Hash        := Hash_function specified in OGA ID
              If hash is not a variable length output hash,
              then en Encode_m, similar to ORCHID Encode_96
              is used
m           := 64

```

4.3.6. Collision risks with Hierarchical HITs

The 64 bit hash size does have an increased risk of collisions over the 96 bit hash size used for the other HIT Suites. There is a 0.01% probability of a collision in a population of 66 million. The probability goes up to 1% for a population of 663 million. See Appendix A for the collision probability formula.

However, this risk of collision is within a single HDA. Further, all HDAs are expected to provide a registration process for reverse lookup validation. This registration process would reject a collision, forcing the client to generate a new HI and thus hierarchical HIT and reapplying to the registration process.

5. IANA Considerations

Because HHIT use of ORCHIDv2 format is not compatible with [RFC7343], IANA is requested to allocated a new 28-bit prefix out of the IANA IPv6 Special Purpose Address Block, namely 2001:0000::/23, as per [RFC6890].

6. Security Considerations

A 64 bit hash space presents a real risk of second pre-image attacks. The HHIT Registry services effectively block attempts to "take over" a HHIT. It does not stop a rogue attempting to impersonate a known HHIT. This attack can be mitigated by the Responder using DNS to find the HI for the HHIT or the RVS for the HHIT that then provides the registered HI.

Another mitigation of HHIT hijacking is if the HI owner supplies an object containing the HHIT and signed by the HI private key of the HDA.

The two risks with hierarchical HITs are the use of an invalid HID and forced HIT collisions. The use of the "hhit.arpa." DNS zone is a strong protection against invalid HIDs. Querying an HDA's RVS for a HIT under the HDA protects against talking to unregistered clients. The Registry service has direct protection against forced or accidental HIT hash collisions.

7. Acknowledgments

The RDA/HDA 16/16 bit split, replacing the original 14/18 split was the result of discussions on lookup and implementation challenges of byte boundaries over nibble boundaries.

The initial versions of this document were developed with the assistance of Xiaohu Xu and Bingyang Liu of Huawei.

Sue Hares contributed to the clarity in this document.

8. References

8.1. Normative References

[new-orchid]

Moskowitz, R., Card, S., and A. Wiethuechter, "Using cSHAKE in ORCHIDs", Work in Progress, Internet-Draft, draft-moskowitz-orchid-cshake-00, 11 December 2019, <<https://tools.ietf.org/html/draft-moskowitz-orchid-cshake-00>>.

[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>>.

[RFC7401] Moskowitz, R., Ed., Heer, T., Jokela, P., and T. Henderson, "Host Identity Protocol Version 2 (HIPv2)", RFC 7401, DOI 10.17487/RFC7401, April 2015, <<https://www.rfc-editor.org/info/rfc7401>>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

8.2. Informative References

- [draft.moskowitz-hip-arch-02]
Moskowitz, R., "Host Identity Payload", Superseded Internet-Draft, draft-moskowitz-hip-arch-02, 22 February 2001.
- [draft.zhang-hip-hierarchical-parameter-00]
Dacheng, Z. and X. Xiaohu, "Extensions of Host Identity Protocol (HIP) with Hierarchical Information", Abandoned Internet-Draft, draft-zhang-hip-hierarchical-parameter-00, 27 May 2009.
- [hhit-registries]
Moskowitz, R., Card, S., and A. Wiethuechter, "Hierarchical HIT Registries", Work in Progress, Internet-Draft, draft-moskowitz-hip-hhit-registries-02, 9 March 2020, <<https://tools.ietf.org/html/draft-moskowitz-hip-hhit-registries-02>>.
- [RFC6537] Ahrenholz, J., "Host Identity Protocol Distributed Hash Table Interface", RFC 6537, DOI 10.17487/RFC6537, February 2012, <<https://www.rfc-editor.org/info/rfc6537>>.
- [RFC6890] Cotton, M., Vegoda, L., Bonica, R., Ed., and B. Haberman, "Special-Purpose IP Address Registries", BCP 153, RFC 6890, DOI 10.17487/RFC6890, April 2013, <<https://www.rfc-editor.org/info/rfc6890>>.
- [RFC7343] Laganier, J. and F. Dupont, "An IPv6 Prefix for Overlay Routable Cryptographic Hash Identifiers Version 2 (ORCHIDv2)", RFC 7343, DOI 10.17487/RFC7343, September 2014, <<https://www.rfc-editor.org/info/rfc7343>>.
- [RFC8004] Laganier, J. and L. Eggert, "Host Identity Protocol (HIP) Rendezvous Extension", RFC 8004, DOI 10.17487/RFC8004, October 2016, <<https://www.rfc-editor.org/info/rfc8004>>.
- [RFC8005] Laganier, J., "Host Identity Protocol (HIP) Domain Name System (DNS) Extension", RFC 8005, DOI 10.17487/RFC8005, October 2016, <<https://www.rfc-editor.org/info/rfc8005>>.
- [WeakKeys] Heninger, N.H., Durumeric, Z.D., Wustrow, E.W., and J.A.H. Halderman, "Detection of Widespread Weak Keys in Network Devices", August 2012, <<https://factorable.net/weakkeys12.extended.pdf>>.

Appendix A. Calculating Collision Probabilities

The accepted formula for calculating the probability of a collision is:

$$p = 1 - e^{\{-k^2/(2n)\}}$$

P Collision Probability
n Total possible population
k Actual population

Authors' Addresses

Robert Moskowitz
HTT Consulting
Oak Park, MI 48237
United States of America

Email: rgm@labs.htt-consult.com

Stuart W. Card
AX Enterprize
4947 Commercial Drive
Yorkville, NY 13495
United States of America

Email: stu.card@axenterprize.com

Adam Wiethuechter
AX Enterprize
4947 Commercial Drive
Yorkville, NY 13495
United States of America

Email: adam.wiethuechter@axenterprize.com

HIP
Internet-Draft
Updates: 7401, 7402 (if approved)
Intended status: Standards Track
Expires: 3 February 2022

R. Moskowitz
HTT Consulting
S. Card
A. Wiethuechter
AX Enterprize
2 August 2021

New Cryptographic Algorithms for HIP
draft-moskowitz-hip-new-crypto-10

Abstract

This document provides new cryptographic algorithms to be used with HIP. The Edwards Elliptic Curve and the Keccak sponge functions are the main focus. The HIP parameters and processing instructions impacted by these algorithms are defined.

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 <https://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 3 February 2022.

Copyright Notice

Copyright (c) 2021 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Terms and Definitions	3
2.1. Requirements Terminology	3
2.2. Definitions	3
3. HIP Parameter values for new Cryptographic Functions	4
3.1. Elliptic Curves for Diffie-Hellman	4
3.1.1. DIFFIE_HELLMAN	5
3.2. Edward Digital Signature Algorithm for HITs	5
3.2.1. HOST_ID	5
3.2.2. HIT_SUITE_LIST	6
3.3. Hashing in HIP	7
3.3.1. Hashing with the Sponge Functions	7
3.3.2. RHASH	8
3.3.3. HIP_MAC and HIP_MAC2	8
3.4. HIP Cipher	9
3.4.1. HIP_CIPHER	9
3.5. ESP Transform	9
3.5.1. ESP_TRANSFORM	10
4. Generating a HIT from an HI	10
5. HIP KEYMAT Generation	10
5.1. The Keccak KEYMAT	11
5.2. The Xoodyak KEYMAT	11
6. Pseudorandom Function (PRF)	12
7. IANA Considerations	12
8. Security Considerations	12
8.1. Keymat vulnerabilities	12
8.2. KMAC Security as a KDF	13
9. Acknowledgments	13
10. References	13
10.1. Normative References	13
10.2. Informative References	15
Authors' Addresses	16

1. Introduction

This document adds new cryptographic algorithms for HIPv2 [RFC7401] and [RFC7402]. This includes:

- * New elliptic curves for ECDH.
- * The Edwards Elliptic Curve Digital Signature Algorithm (EdDSA) used in Host Identities (HI) and for Base Exchange (BEX) signatures.
- * Hashes used in Host Identity Tag (HIT) generation, and wherever else hashes are needed.

- * Keyed hashes used for KEYMAT generation and packet MACing operations.
- * AEAD and stream ciphers to use in HIP and HIP enabled secure communication protocols.

The hashes and encryption are all built on the Keccak [Keccak] sponge function and the Xoodyak [Xoodyak] lightweight scheme.

These additions reflect selection of advances in the field of cryptography that would best benefit HIP, particularly in constrained devices and communications.

Ed Note: The Xoodyak function calls should be considered the 1st best effort. There are a few areas open for discussion, like which of the 3 choices for adding in the nonce to the AEAD mode and when to use counter and Id. Also there may be copy errors from the source specification, nicer function calls, better acronyms.

2. Terms and Definitions

2.1. Requirements Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2.2. Definitions

cSHAKE (The customizable SHAKE function [NIST SP800-185]):
Extends the SHAKE scheme to allow users to customize their use of the function.

DEC function (Doubly-Extendable Cryptographic function):
An extendable output function (XOF) that accepts sequences of strings as input and that supports incremental queries efficiently.

DECK function (Doubly-Extendable Cryptographic Keyed function):
A keyed function that takes a sequence of input strings and returns a pseudorandom string of arbitrary length and that can be computed incrementally.

Keccak:

The family of all sponge functions with a KECCAK-f permutation as the underlying function and multi-rate padding as the padding rule. In particular all the functions referenced from [NIST FIPS-202] and [NIST SP800-185].

KMAC (KECCAK Message Authentication Code [NIST SP800-185]):

A pseudo random function (PRF) and keyed hash function based on KECCAK.

SHAKE (Secure Hash Algorithm KECCAK [NIST FIPS-202]):

A secure hash that allows for an arbitrary output length. SHAKE128 and SHAKE256 are instances of XOFs. SHAKE is shorthand for SHAKE128.

PRF (Pseudorandom Function):

A function that takes as input a key and that it is hard to distinguish from a random oracle by an adversary that does not know the key.

XHASH (Xoodyak Hash Algorithm):

A secure hash, based on Xoodyak, that allows for an arbitrary output length. XHASH is an instance of XOF.

XMAC (Xoodyak Message Authentication Code):

A keyed hash function similar to KMAC, based on Xoodyak, that allows for an arbitrary output length.

XOF (eXtendable-Output Function [NIST FIPS-202]):

A function on bit strings (also called messages) in which the output can be extended to any desired length.

3. HIP Parameter values for new Cryptographic Functions

HIP parameters carry information that is necessary for establishing and maintaining a HIP association. For example, the device's public keys as well as the signaling for negotiating ciphers and payload handling are encapsulated in HIP parameters. Additional information, meaningful for end hosts or middleboxes, may also be included in HIP parameters. The specification of the HIP parameters and their mapping to HIP packets and packet types is flexible to allow HIP extensions to define new parameters and new protocol behavior.

3.1. Elliptic Curves for Diffie-Hellman

Elliptic curves Curve25519 and Curve448 [RFC7748] are specified here for use in the HIP Diffie-Hellman exchange.

Curve25519 and Curve448 are already defined in Section 5.2.1 of [hip-dex], using the HIP-DEX CKDF. Here they are defined for using the new KMAC [NIST SP800-185] or XMAC [Xoodyak] derived KDF in Section 5.

3.1.1. DIFFIE_HELLMAN

The DIFFIE_HELLMAN parameter may be included in selected HIP packets based on the DH Group ID selected. The DIFFIE_HELLMAN parameter is defined in Section 5.2.7 of [RFC7401].

The following Elliptic Curves are defined here:

Group	KDF	Value
Curve25519 [RFC7748]	KMAC	13
Curve448 [RFC7748]	KMAC	14

A new KDF for KEYMAT, Section 6.5 of [RFC7401] using Keccak or Xoodyak is defined in Section 5.

3.2. Edward Digital Signature Algorithm for HITs

This section is extracted from Appendix D of [drip-rid]. It may later be pulled and only maintained there.

Edwards-Curve Digital Signature Algorithm (EdDSA) [RFC8032] are specified here for use as Host Identities (HIs) per HIPv2 [RFC7401]. Further the HIT_SUITE_LIST is specified as used in [RFC7343].

3.2.1. HOST_ID

The HOST_ID parameter specifies the public key algorithm, and for elliptic curves, a name. The HOST_ID parameter is defined in Section 5.2.19 of [RFC7401].

Algorithm profiles	Values
EdDSA	13 [RFC8032]

For hosts that implement EdDSA as the algorithm, the following ECC curves are available:

Algorithm	Curve	Values
EdDSA	RESERVED	0
EdDSA	EdDSA25519	1 [RFC8032]
EdDSA	EdDSA25519ph	2 [RFC8032]
EdDSA	EdDSA448	3 [RFC8032]
EdDSA	EdDSA448ph	4 [RFC8032]

3.2.2. HIT_SUITE_LIST

The HIT_SUITE_LIST parameter contains a list of the supported HIT suite IDs of the Responder. Based on the HIT_SUITE_LIST, the Initiator can determine which source HIT Suite IDs are supported by the Responder. The HIT_SUITE_LIST parameter is defined in Section 5.2.10 of [RFC7401].

The following HIT Suite ID is defined, and the relationship between the four-bit ID value used in the OGA ID field and the eight-bit encoding within the HIT_SUITE_LIST ID field is clarified:

HIT Suite	Four-bit ID	Eight-bit encoding
RESERVED	0	0x00
EdDSA/cSHAKE128	5	0x50
EdDSA/XHASH	6	0x60

The following table provides more detail on the above HIT Suite combinations. The input for each generation algorithm is the encoding of the HI as defined herein.

The output of cSHAKE128 and XHASH are variable per the needs of a specific ORCHID construction. It is at most 96 bits long and is directly used in the ORCHID (without truncation).

Index	Hash function	HMAC	Signature algorithm family	Description
5	cSHAKE128	KMAC128	EdDSA	EdDSA HI hashed with cSHAKE128, output is variable
6	XHASH	XMAC	EdDSA	EdDSA HI hashed with XMAC, output is variable

Table 1: HIT Suites

3.3. Hashing in HIP

Hashing is used in HIP for HIT generation and keyed hashes of HIP payloads. The hash algorithm used is designated as part of the HIT_SUITE_ID. The keyed hash function is the "common" such function used in conjunction with the HIT hash.

3.3.1. Hashing with the Sponge Functions

The XOF function in SHA-3, Secure Hash Algorithm Keccak (SHAKE) [NIST FIPS-202] and the more recent Xoodyak [Xoodyak] algorithm are called sponge functions. Sponge functions have a special feature in which an arbitrary number of output bits are "squeezed" out of the hashing state. This is a significant use change in that hash truncation or multiple "runs" for enough bits are not used with sponge functions.

3.3.1.1. cSHAKE, the customizable SHAKE function

The customizable SHAKE function (cSHAKE) in [NIST SP800-185] will be used as a HIP hash. As a Keccak XOF, it does not use the truncation operation that other hashes need. The invocation of cSHAKE specifies the desired number of bits in the hash output. Further, cSHAKE has a parameter 'S' as a customization bit string. This parameter will be used for including hash specific customization like the ORCHID Context Identifier in a standard fashion.

Hardware implementation of Keccak in VHDL is available from Keccak [Keccak] team website.

3.3.1.2. The Xoodyak Hash

The Xoodyak [Xoodyak] sponge function is a candidate in the NIST Lightweight Cryptography (LWC) Standardization process (see [NISTIR 8369]). Xoodyak has been selected here for use in HIP from the LWC 2nd round candidates as it was developed by the Keccak team, making it more directly in line with Keccak.

Xoodyak has a hash function mode. More specifically, this hash mode is an extendable output function (XOF).

As the Xoodyak specification [Xoodyak_Spec] does not provide high-level function calls, rather a set of primitives to use to construct the various modes, the appropriate primitive calls will be detailed below. Xoodyak as a hash will be called here "XHASH".

To get a n-byte digest of some input x: XHASH(n, x), use the following set of Xoodyak primitives:

```
Cyclist(,,)
Absorb(x)
Squeeze(n)
```

Xoodyak can also naturally implement a DEC function and process a sequence of strings. Here the output depends on the sequence as such and not just on the concatenation of the different strings. To compute a n-byte digest, XHASH(n, {x1, x2, x3}) the Xoodyak primitives are:

```
Cyclist(,,)
Absorb(x1)
Absorb(x2)
Absorb(x3)
Squeeze(n)
```

The equivalent of the parameter 'S' in cSHAKE above can be implemented as the last Absorb primitive call in the DEC function. That is: XHASH(L, {S, N, X}) is equivalent to cSHAKE(X, L, N, S).

3.3.2. RHASH

RHASH is the general term used throughout [RFC7401] to refer to the hash used for a specific HIT suite. For this addendum cSHAKE128 for Keccak or XHASH for Xoodyak is used, even for HITs of EdDSA448.

Unless otherwise specified, L of cSHAKE128 or n of XHASH is 256, resulting in a similar output to SHA256. Any truncation used for, older, fixed output hashes is still used. This is to simplify code integration. One exception to this is in Section 4.

3.3.3. HIP_MAC and HIP_MAC2

The HIP_MAC and HIP_MAC2 parameters in [RFC7401] use HMAC [RFC2104]. This performs two hashes on a string with a key for a keyed hash the length of the underlying hash.

For both HIP_MAC and HIP_MAC2 use, the parameter S below is NULL. It is included for complete function definition.

3.3.3.1. The Keccak Keyed MAC

Here, KMAC from NIST SP 800-185 [NIST SP800-185] is used. This is a single pass using the underlying cSHAKE function. The function call is:

```
KMAC128(Key, Input String, 256, S)
```

3.3.3.2. The Xoodoo Keyed MAC

Here, XMAC is defined as the keyed hash function based on Xoodoo. It is built with primitives from [Xoodoo_Spec] as a DEC function.

To get a n-byte keyed MAC of some input x: XMAC(Key, n, {x, S}). Where n=256, use the following set of Xoodoo primitives:

```
Cyclist(Key, Id,)
Absorb(S)           Only if S is non-null
Absorb(Input String)
Squeeze(32)
```

Id is "HIP_MAC" and "HIP_MAC2" respectively. Note since S is null in this XMAC usage, the first Absorb call is not performed.

3.4. HIP Cipher

HIP encrypted parameters use the HIP_CIPHER, Section 5.2.8 of [RFC7401]. The Xoodoo cipher, [Xoodoo], is recommended. Here Xoodoo is used in encrypt only mode.

3.4.1. HIP_CIPHER

The HIP_CIPHER parameter value for Xoodoo is:

hip_cipher Suite ID	Value
Xoodoo	6 (Xoodoo)

The Xoodoo primitive calls for encrypt only are:

```
Cyclist(Key, Id,)
Absorb(IV)
C Encrypt(P)
```

Where Id is HIP parameter name (e.g. "ENCRYPTED").
 IV is from the encrypted HIP parameter.
 P is the plain-text per the specific HIP encrypted parameter.
 C is the ciphertext.

3.5. ESP Transform

The ESP_TRANSFORM parameter is used during ESP SA establishment, Section 5.1.2 of [RFC7402]. The Xoodoo cipher, [Xoodoo], is recommended. Here Xoodoo is used in AEAD mode.

Further, it is recommended to use Implicit IV ESP [RFC8750] to match its lightweight over-the-air format with the lightweight Xoodoo AEAD cipher.

3.5.1. ESP_TRANSFORM

The ESP_TRANSFORM Suite IDs for Xoodoo are:

hip_cipher			
Suite ID	Value		
Xoodoo-96	16	(Xoodoo)	
Xoodoo	17	(Xoodoo)	
Implicit IV	18	[8750]	

The Implicit IV Suite ID is unique in that it is an AND condition with ciphers that can use it. That is AES-GCM and Xoodoo can both use 'regular' ESP [RFC4303] or [RFC8750].

The Xoodoo primitive calls for AEAD encrypt are:

```
Cyclist(Key, Id,)
Absorb(IV)
Absorb(A)
C   Encrypt(P)
T   Squeeze(t)
```

Where Id is "ESP_TRANSFORM". The IV is either a 32 bit ESP IV per [RFC4303] or the ESP Seq Number per [RFC8750]. P is the plain-text and A is the associated data. t is either 12 or 16. T is the ESP ICV of length t.

4. Generating a HIT from an HI

The EdDSA/cSHAKE based HITs require a new ORCHID generation method than that described in section 3.2 of [RFC7401]. The XOF functionality of cSHAKE produces an output of L bits. This replaces the Encode_96 function in the ORCHID generation.

For identities that are EdDSA public keys, ORCHIDs will be generated per the process defined in Appendix C.2.1 of [drip-rid].

5. HIP KEYMAT Generation

For either the Keccak or Xoodoo KEYMAT generation, the inputs are consistent. The only practical difference is that cSHAKE allows for 128 or 256 bits of strength, whereas Xoodoo only provides 128 bits.

L is the derived key bit length. Since 4 HIP keys are "drawn" from this output, the length is $4 * \text{HIP_key_size}$. Per ASIACRYPT 2017, pp. 606-637 [ASIACRYPT-2017] each of these derived keys will have the same strength as the Diffie-Hellman shared secret.

S is the byte string 01001011 || 01000100 || 01000110, which represents the sequence of characters "K", "D", and "F" in 8-bit ASCII.

Salt and info are derived as defined in sec 6.5 of [RFC7401]. There are special security considerations for IKM per [RFC7748].

5.1. The Keccak KEYMAT

The KMAC function provides a new, more efficient, key derivation function over HKDF [RFC5869]. KMAC as a KDF is defined below.

The two HIs MUST be used in constructing IKM as follows:

$$\text{IKM} = \text{Diffie-Hellman secret} \mid \text{sort}(\text{HI-I} \mid \text{HI-R})$$

The two HIs are separately DER encoded per [RFC7401]

The choice of KMAC128 or KMAC256 is based on the strength of the output key material. For 256 bits of strength equivalent to HMAC-SHA256, use KMAC256. Per [NIST SP800-56Cr1], Section 4.1, Option 3:

$$\text{OKM} = \text{KMAC}[128|256](\text{salt} \mid \text{info}, \text{IKM}, \text{L}, \text{S})$$

5.2. The Xoodyak KEYMAT

Here, XMAC from Section 3.3.3.2 is used. The DEC function $\text{XMAC}("", \text{L}, \{\text{DH}, \text{sort}(\text{HI-I}, \text{HI-R}), \text{info}, \text{Salt}, \text{S}\})$ primitives are:

```
Cyclist( , , )
Absorb(S)
Absorb(salt)
Absorb(info)
Absorb(max(HI-I , HI-R))
Absorb(min(HI-I , HI-R))
Absorb(Diffie-Hellman secret)
Squeeze(L)   Where L is bytes
```

Ed Note: Need to check that all above are well defined bytestrings per 7401. I think they are.

6. Pseudorandom Function (PRF)

Appendix B of NIST SP 800-185 [NIST SP800-185] defines how to use SHAKE, cSHAKE, or KMAC as a PRF.

For Xoodyak, XMAC from Section 3.3.3.2 is used in the same manner as KMAC above.

7. IANA Considerations

IANA will need to make the following changes to the "Host Identity Protocol (HIP) Parameters" registries:

Diffie Hellman:

This document defines the new Curve25519 and Curve448 for the Diffie-Hellman exchange (see Section 3.1.1).

Host ID:

This document defines the new EdDSA Host ID (see Section 3.2.1).

HIT Suite ID:

This document defines the new HIT Suite of EdDSA/cSHAKE and EdDSA/XHASH (see Section 3.2.2).

HIP Cipher:

This document defines the new Xoodyak cipher for HIP encrypted parameters (see Section 3.4.1).

ESP Transform:

This document defines the new Xoodyak cipher and use of [RFC8750] for the ESP Transform parameter (see Section 3.5).

8. Security Considerations

8.1. Keymat vulnerabilities

[RFC7748] warns about using Curve25519 and Curve448 in Diffie-Hellman for key derivation:

Designers using these curves should be aware that for each public key, there are several publicly computable public keys that are equivalent to it, i.e., they produce the same shared secrets. Thus using a public key as an identifier and knowledge of a shared secret as proof of ownership (without including the public keys in the key derivation) might lead to subtle vulnerabilities.

Thus the two Host IDs are included with the Diffie-Hellman secret in the KEYMAT generation.

8.2. KMAC Security as a KDF

Section 4.1 of NIST SP 800-185 [NIST SP800-185] states:

"The KECCAK Message Authentication Code (KMAC) algorithm is a PRF and keyed hash function based on KECCAK . It provides variable-length output"

That is, the output of KMAC is indistinguishable from a random string, regardless of the length of the output. As such, the output of KMAC can be divided into multiple substrings, each with the strength of the function (KMAC128 or KMAC256) and provided that a long enough key is used, as discussed in Sec. 8.4.1 of SP 800-185.

For example KMAC128(K, X, 512, S), where K is at least 128 bits, can produce 4 128 bit keys each with a strength of 128 bits. That is a single sponge operation is replacing perhaps 5 HMAC-SHA256 operations (each 2 SHA256 operations) in HKDF.

9. Acknowledgments

Quynh Dang of NIST gave considerable guidance on using Keccak and the NIST supporting documents. Joan Deamen of the Keccak team was especially helpful in many aspects of using Keccak and Xoodyak, particularly with the KEYMAT section and the strength of the derived keys.

NIST is entering round 3 (final) of its Lightweight Crypto Competition with anticipated selection the end of 2021 or early in 2022. Events in this process will impact selections in this document.

10. References

10.1. Normative References

[NIST FIPS-202]

Dworkin, M., "SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions", National Institute of Standards and Technology report, DOI 10.6028/nist.fips.202, July 2015, <<https://doi.org/10.6028/nist.fips.202>>.

[NIST SP800-185]

Kelsey, J., Change, S., and R. Perlner, "SHA-3 derived functions: cSHAKE, KMAC, TupleHash and ParallelHash", National Institute of Standards and Technology report, DOI 10.6028/nist.sp.800-185, December 2016, <<https://doi.org/10.6028/nist.sp.800-185>>.

[NIST SP800-56Cr1]

Barker, E., Chen, L., and R. Davis, "Recommendation for key-derivation methods in key-establishment schemes", National Institute of Standards and Technology report, DOI 10.6028/nist.sp.800-56cr1, April 2018, <<https://doi.org/10.6028/nist.sp.800-56cr1>>.

[NISTIR 8369]

Sonmez Turan, M., McKay, K., Chang, D., Calik, C., Bassham, L., Kang, J., and J. Kelsey, "Status Report on the Second Round of the NIST Lightweight Cryptography Standardization Process", National Institute of Standards and Technology report, DOI 10.6028/nist.ir.8369, July 2021, <<https://doi.org/10.6028/nist.ir.8369>>.

[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>>.

[RFC7401] Moskowitz, R., Ed., Heer, T., Jokela, P., and T. Henderson, "Host Identity Protocol Version 2 (HIPv2)", RFC 7401, DOI 10.17487/RFC7401, April 2015, <<https://www.rfc-editor.org/info/rfc7401>>.

[RFC7402] Jokela, P., Moskowitz, R., and J. Melen, "Using the Encapsulating Security Payload (ESP) Transport Format with the Host Identity Protocol (HIP)", RFC 7402, DOI 10.17487/RFC7402, April 2015, <<https://www.rfc-editor.org/info/rfc7402>>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

[Xoodyak] Daemen, J., Hoffert, S., Peeters, M., Van Assche, G., and R. Van Keer, "The Xoodyak Cipher and Hash", <<https://keccak.team/xoodyak.html>>.

[Xoodyak_Spec]

Daemen, J., Hoffert, S., Peeters, M., Van Assche, G., and R. Van Keer, "Xoodoo cookbook", 2019, <<https://eprint.iacr.org/2018/767.pdf>>.

10.2. Informative References

[ASIACRYPT-2017]

Daemen, J., Mennink, B., and G. Van Assche, "Full-State Keyed Duplex with Built-In Multi-user Support", DOI 10.1007/978-3-319-70697-9_21, Advances in Cryptology - ASIACRYPT 2017 pp. 606-637, 2017, <https://doi.org/10.1007/978-3-319-70697-9_21>.

[drip-rid]

Moskowitz, R., Card, S. W., Wiethuechter, A., and A. Gurtov, "Unmanned Aircraft System Remote Identification (UAS RID)", Work in Progress, Internet-Draft, draft-ietf-drip-rid-08, 25 July 2021, <<https://tools.ietf.org/html/draft-ietf-drip-rid-08>>.

[hip-dex]

Moskowitz, R., Hummen, R., and M. Komu, "HIP Diet EXchange (DEX)", Work in Progress, Internet-Draft, draft-ietf-hip-dex-24, 19 January 2021, <<https://tools.ietf.org/html/draft-ietf-hip-dex-24>>.

[Keccak]

Bertoni, G., Daemen, J., Peeters, M., Van Assche, G., and R. Van Keer, "The Keccak Function", <<https://keccak.team/index.html>>.

[RFC2104]

Krawczyk, H., Bellare, M., and R. Canetti, "HMAC: Keyed-Hashing for Message Authentication", RFC 2104, DOI 10.17487/RFC2104, February 1997, <<https://www.rfc-editor.org/info/rfc2104>>.

[RFC4303]

Kent, S., "IP Encapsulating Security Payload (ESP)", RFC 4303, DOI 10.17487/RFC4303, December 2005, <<https://www.rfc-editor.org/info/rfc4303>>.

[RFC5869]

Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, DOI 10.17487/RFC5869, May 2010, <<https://www.rfc-editor.org/info/rfc5869>>.

[RFC7343]

Laganier, J. and F. Dupont, "An IPv6 Prefix for Overlay Routable Cryptographic Hash Identifiers Version 2 (ORCHIDv2)", RFC 7343, DOI 10.17487/RFC7343, September 2014, <<https://www.rfc-editor.org/info/rfc7343>>.

- [RFC7748] Langley, A., Hamburg, M., and S. Turner, "Elliptic Curves for Security", RFC 7748, DOI 10.17487/RFC7748, January 2016, <<https://www.rfc-editor.org/info/rfc7748>>.
- [RFC8032] Josefsson, S. and I. Liusvaara, "Edwards-Curve Digital Signature Algorithm (EdDSA)", RFC 8032, DOI 10.17487/RFC8032, January 2017, <<https://www.rfc-editor.org/info/rfc8032>>.
- [RFC8750] Migault, D., Guggemos, T., and Y. Nir, "Implicit Initialization Vector (IV) for Counter-Based Ciphers in Encapsulating Security Payload (ESP)", RFC 8750, DOI 10.17487/RFC8750, March 2020, <<https://www.rfc-editor.org/info/rfc8750>>.

Authors' Addresses

Robert Moskowitz
HTT Consulting
Oak Park, MI 48237
United States of America

Email: rgm@labs.htt-consult.com

Stuart W. Card
AX Enterprize
4947 Commercial Drive
Yorkville, NY 13495
United States of America

Email: stu.card@axenterprize.com

Adam Wiethuechter
AX Enterprize
4947 Commercial Drive
Yorkville, NY 13495
United States of America

Email: adam.wiethuechter@axenterprize.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 4, 2020

E. Kinnear
T. Pauly
C. Wood
Apple Inc.
P. McManus
Fastly
November 01, 2019

Adaptive DNS: Improving Privacy of Name Resolution
draft-pauly-dprive-adaptive-dns-privacy-01

Abstract

This document defines an architecture that allows clients to dynamically discover designated resolvers that offer encrypted DNS services, and use them in an adaptive way that improves privacy while co-existing with locally provisioned resolvers. These resolvers can be used directly when looking up names for which they are designated. These resolvers also provide the ability to proxy encrypted queries, thus hiding the identity of the client requesting resolution.

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 <https://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, 2020.

Copyright Notice

Copyright (c) 2019 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents

carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
1.1. Specification of Requirements	4
2. Terminology	4
3. Client Behavior	5
3.1. Discovering Designated DoH Servers	6
3.1.1. Whitelisting Designated DoH Servers	7
3.1.2. Accessing Extended Information	8
3.2. Discovering Local Resolvers	9
3.3. Hostname Resolution Algorithm	10
3.4. Oblivious Resolution	11
3.5. Handling Network Changes	12
4. Server Requirements	12
4.1. Provide a DoH Server	12
4.1.1. Oblivious DoH Proxy	12
4.1.2. Oblivious DoH Target	13
4.1.3. Keying Material	13
4.2. Advertise the DoH Server	13
4.3. Provide Extended Configuration as a Web PvD	13
5. Server Deployment Considerations	15
5.1. Single Content Provider	15
5.2. Multiple Content Providers	15
5.3. Avoid Narrow Deployments	16
6. Local Resolver Deployment Considerations	16
6.1. Designating Local DoH Servers	16
6.2. Local Use Cases	17
6.2.1. Accessing Local-Only Resolvable Content	17
6.2.2. Accessing Locally Optimized Content	18
6.2.3. Walled-Garden and Captive Network Deployments	19
6.2.4. Network-Based Filtering	19
7. Performance Considerations	20
8. Security Considerations	21
9. Privacy Considerations	21
10. IANA Considerations	22
10.1. DoH Template PvD Key	22
10.2. DNS Filtering PvD Keys	22
10.3. DoH URI Template DNS Parameter	23
11. Acknowledgments	23
12. References	23
12.1. Normative References	23
12.2. Informative References	24

Authors' Addresses	25
------------------------------	----

1. Introduction

When clients need to resolve names into addresses in order to establish networking connections, they traditionally use by default the DNS resolver that is provisioned by the local network along with their IP address [RFC2132] [RFC8106]. Alternatively, they can use a resolver indicated by a tunneling service such as a VPN.

However, privacy-sensitive clients might prefer to use an encrypted DNS service other than the one locally provisioned in order to prevent interception, profiling, or modification by entities other than the operator of the name service for the name being resolved. Protocols that can improve the transport security of a client when using DNS or creating TLS connections include DNS-over-TLS [RFC7858], DNS-over-HTTPS [RFC8484], and encrypted Server Name Indication (ESNI) [I-D.ietf-tls-esni].

There are several concerns around a client using such privacy-enhancing mechanisms for generic system traffic. A remote service that provides encrypted DNS may not provide correct answers for locally available resources, or private resources (such as domains only accessible over a private network). Remote services may also be untrusted from a privacy perspective: while encryption will prevent on-path observers from seeing hostnames, client systems need to trust the encrypted DNS service to not store or misuse queries made to it. Further, extensive use of cloud based recursive resolvers obscures the network location of the client which may degrade the performance of the returned server due to lack of proximity at the benefit of improved privacy.

Client systems are left with choosing between one of the following stances:

1. Send all application DNS queries to a particular encrypted DNS service, which requires establishing user trust of the service. This can lead to resolution failures for local or private enterprise domains absent heuristics or other workarounds for detecting managed networks.
2. Allow the user or another entity to configure local policy for which domains to send to local, private, or encrypted resolvers. This provides more granularity at the cost of increasing user burden.
3. Only use locally-provisioned resolvers, and opportunistically use encrypted DNS to these resolvers when possible. (Clients may

learn of encrypted transport support by actively probing such resolvers.) This provides marginal benefit over not using encrypted DNS at all, especially if clients have no means of authenticating or trusting local resolvers.

This document defines an architecture that allows clients to improve the privacy of their DNS queries without requiring user intervention, and allowing coexistence with local, private, and enterprise resolvers.

This architecture is composed of several mechanisms:

- o A DNS record that indicates a designated DoH server associated with a name (Section 3.1);
- o an extension to DoH that allows client IP addresses to be disassociated from queries via proxying ([I-D.pauly-dprive-oblivious-doh]);
- o a DoH server that responds to queries directly and supports proxying (Section 4);
- o and client behavior rules for how to resolve names using a combination of designated DoH resolvers, proxied queries, and local resolvers (Section 3).

1.1. Specification of Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Terminology

This document defines the following terms:

Adaptive DNS: Adaptive DNS is a technique to provide an encrypted transport for DNS queries that can be sent directly to a Designated DoH Server, to use Oblivious DoH to hide the client IP address, or to use Direct Resolvers when required or appropriate.

Designated DoH Server: A DNS resolver that provides connectivity over HTTPS (DoH) that is designated as a responsible resolver for a given domain or zone.

Direct Resolver: A DNS resolver using any transport that is provisioned directly by a local router or a VPN.

Exclusive Direct Resolver: A Direct Resolver that requires the client to use it exclusively for a given set of domains, such as private domains managed by a VPN. This status is governed by local system policy.

Oblivious DoH: A technique that uses multiple DoH servers to proxy queries in a way that disassociates the client's IP address from query content.

Oblivious Proxy: A resolution server that proxies encrypted client DNS queries to another resolution server that will be able to decrypt the query (the Oblivious Target).

Oblivious Target: A resolution server that receives encrypted client DNS queries via an Oblivious Proxy.

Privacy-Sensitive Connections: Connections made by clients that are explicitly Privacy-Sensitive are treated differently from connections made for generic system behavior, such as non-user-initiated maintenance connections. This distinction is only relevant on the client, and does not get communicated to other network entities. Certain applications, such as browsers, can choose to treat all connections as privacy-sensitive.

Web PvD: A Web Provisioning Domain, or Web PvD, represents the configuration of resolvers, proxies, and other information that a server deployment makes available to clients. See Section 4.3.

3. Client Behavior

Adaptive DNS allows client systems and applications to improve the privacy of their DNS queries and connections, both by requiring confidentiality via encryption, and by limiting the ability to correlate client IP addresses with query contents. Specifically, the goal for client queries is to achieve the following properties:

1. No party other than the client and server can learn or control the names being queried by the client or the answers being returned by the server.
2. Only a designated DNS resolver associated with the deployment that is also hosting content will be able to read both the client IP address and queried names for Privacy-Sensitive Connections. For example, a resolver owned and operated by the same provider that hosts "example.com" would be able to link queries for

"example.com" to specific clients (by their IP address), since the server ultimately has this capability once clients subsequently establish secure (e.g., TLS) connections to an address to which "example.com" resolves.

3. Clients will be able to comply with policies required by VPNs and local networks that are authoritative for private domains.

An algorithm for determining how to resolve a given name in a manner that satisfies these properties is described in Section 3.3. Note that this algorithm does not guarantee that responses that are not signed with DNSSEC are valid, and clients that establish connections to unsigned addresses may still expose their local IP addresses to attackers that control their terminal resolver even if hidden during resolution.

3.1. Discovering Designated DoH Servers

All direct (non-oblivious) queries for names in privacy-sensitive connections MUST be sent to a server that both provides encryption and is designated for the domain.

Clients dynamically build and maintain a set of known Designated DoH Servers. The information that is associated with each server is:

- o The URI Template of the DoH server [RFC8484]
- o The public HPKE [I-D.irtf-cfrg-hpke] key of the DoH server used for proxied oblivious queries [I-D.pauly-dprive-oblivious-doh]
- o A list of domains for which the DoH server is designated

This information can be retrieved from several different sources. The primary source for discovering Designated DoH Server configurations is from properties stored in a SVCB (or a SVCB-conformant type like HTTPSSVC) DNS Record [I-D.nygren-dnsop-svc-httpssvc]. This record provides the URI Template and the public Oblivious DoH key of a DoH server that is designated for a specific domain. A specific domain may have more than one such record.

In order to designate a DoH server for a domain, a SVCB record can contain the "dohuri" (Section 10). The value stored in the parameter is a URI, which is the DoH URI template [RFC8484].

The public key of the DoH server is sent as the "odohkey" [I-D.pauly-dprive-oblivious-doh].

The following example shows a record containing a DoH URI, as returned by a query for the HTTPSSVC variant of the SVCB record type on "example.com".

```
example.com.      7200  IN HTTPSSVC 0 svc.example.net.  
svc.example.net. 7200  IN HTTPSSVC 2 svc1.example.net. (  
                                dohuri=https://doh.example.net/dns-query  
                                odohkey="..." )
```

Clients MUST ignore any DoH server URI that was not retrieved from a DNSSEC-signed record that was validated by the client [RFC4033].

Whenever a client resolves a name for which it does not already have a Designated DoH Server, it SHOULD try to determine the Designated DoH Server by sending a query for the an SVCB record for the name. If there is no DoH server designated for the name or zone, signaled either by an NXDOMAIN answer or a SVCB record that does not contain a DoH URI, the client SHOULD suppress queries for the SVCB record for a given name until the time-to-live of the answer expires.

In order to bootstrap discovery of Designated DoH Servers, client systems SHOULD have some saved list of at least two names that they use consistently to perform SVCB record queries on the Direct Resolvers configured by the local network. Since these queries are likely not private, they SHOULD NOT be associated with user action or contain user-identifying content. Rather, the expectation is that all client systems of the same version and configuration would issue the same bootstrapping queries when joining a network for the first time when the list of Designated DoH Servers is empty.

3.1.1. Whitelisting Designated DoH Servers

Prior to using a Designated DoH Server for direct name queries on privacy-sensitive connections, clients MUST whitelist the server.

The requirements for whitelisting are:

- o Support for acting as an Oblivious Proxy. Each Designated DoH Server is expected to support acting as a proxy for Oblivious DoH. A client MUST issue at least one query that is proxied through the server before sending direct queries to the server.
- o Support for acting as an Oblivious Target. Each Designated DoH Server is expected to support acting as a target for Oblivious DoH. A client MUST issue at least one query that is targeted at the server through a proxy before sending direct queries to the server.

Designated DoH Servers are expected to act both as Oblivious Proxies and as Oblivious Targets to ensure that clients have sufficient options for preserving privacy using Oblivious DoH. Oblivious Targets are expected to act as Oblivious Proxies to ensure that no Oblivious DoH server can act as only a target (thus being able to see patterns in name resolution, which might have value to a resolver) and require other servers to take on a disproportionate load of proxying.

Clients MAY further choose to restrict the whitelist by other local policy. For example, a client system can have a list of trusted resolver configurations, and it can limit the whitelist of Designated DoH Servers to configurations that match this list. Alternatively, a client system can check a server against a list of audited and approved DoH Servers that have properties that the client approves.

Clients SHOULD NOT whitelist authority mappings for effective top-level domains (eTLDs), such as ".com".

If a client detects at any point after whitelisting a DoH server that the server no longer meets the criteria for whitelisting, such as consistently failing to proxy or receive Oblivious DoH queries, the client SHOULD remove the DoH server from its whitelist.

3.1.2. Accessing Extended Information

When a Designated DoH Server is discovered, clients SHOULD also check to see if this server provides an extended configuration in the form of a Web PvD (Section 4.3). To do this, the client performs a GET request to the DoH URI, indicating that it accepts a media type of "application/pvd+json" [I-D.ietf-intarea-provisioning-domains]. When requesting the PvD information, the query and fragment components of the requested path are left empty. Note that this is different from a GET request for the "application/dns-message" type, in which the query variable "dns" contains an encoded version of a DNS message.

In response, the server will return the JSON content for the PvD, if present. The content-type MUST be "application/pvd+json".

The following exchange shows an example of a client retrieving a Web PvD configuration for a DoH server with the URI Template "https://dnsserver.example.net/dns-query".

The client sends:

```
:method = GET
:scheme = https
:authority = dnsserver.example.net
:path = /dns-query
accept = application/pvd+json
```

And the server replies:

```
:status = 200
content-type = application/pvd+json
content-length = 175
cache-control = max-age=86400
```

<JSON content of the Web PvD>

If the server does not support retrieving any extended PvD information, it MUST reply with HTTP status code 415 (Unsupported Media Type, [RFC7231]).

If the retrieved JSON contains a "dnsZones" array [I-D.ietf-intarea-provisioning-domains], the client SHOULD perform an SVCB record lookup of each of the listed zones on the DoH server and validate that the DoH server is a designated server for the domain; and if it is, add the domain to the local configuration.

3.2. Discovering Local Resolvers

If the local network provides configuration with an Explicit Provisioning Domain (PvD), as defined by [I-D.ietf-intarea-provisioning-domains], clients can learn about domains for which the local network's resolver is authoritative.

If an RA provided by the router on the network defines an Explicit PvD that has additional information, and this additional information JSON dictionary contains the key "dohTemplate" (Section 10), then the client SHOULD add this DoH server to its list of known DoH configurations. The domains that the DoH server claims authority for are listed in the "dnsZones" key. Clients MUST use an SVCB record from the locally-provisioned DoH server and validate the answer with DNSSEC [RFC4033] before creating a mapping from the domain to the server. Once this has been validated, clients can use this server for resolution as described in step 2 of Section 3.3.

See Section 6 for local deployment considerations.

3.3. Hostname Resolution Algorithm

When establishing a secure connection to a certain hostname, clients need to first determine which resolver configuration ought to be used for DNS resolution.

Several of the steps outlined in this algorithm take into account the success or failure of name resolution. Failure can be indicated either by a DNS response, such as SERVFAIL or NXDOMAIN, or by a connection-level failure, such as a TCP reset, TLS handshake failure, or an HTTP response error status. In effect, any unsuccessful attempt to resolve a name can cause the client to try another resolver if permitted by the algorithm. This is particularly useful for cases in which a name may not be resolvable over public DNS but has a valid answer only on the local network.

Given a specific hostname, the order of preference for which resolver to use SHOULD be:

1. An Exclusive Direct Resolver, such as a resolver provisioned by a VPN, with domain rules that include the hostname being resolved. If the resolution fails, the connection will fail. See Section 3.2 and Section 6.
2. A Direct Resolver, such as a local router, with domain rules that are known to be authoritative for the domain containing the hostname. If the resolution fails, the connection will try the next resolver configuration based on this list.
3. The most specific Designated DoH Server that has been whitelisted (Section 3.1.1) for the domain containing the hostname, i.e., the designated DoH server which is associated with the longest matching suffix of the hostname. For example, given two Designated DoH Servers, one for "foo.example.com" and another "example.com", clients connecting to "bar.foo.example.com" should use the former. Note that the matching MUST be performed on entire labels. That is, if "example.com" has a designated DoH server, it can be used for "foo.example.com", but not for "badexample.com". If the resolution fails, the connection can either use an Oblivious DoH resolver (Step 4) or the default resolver (Step 5). Privacy-Sensitive Connections SHOULD NOT skip Step 4. Other connections MAY skip Step 4, based on system policy.
4. Oblivious DoH queries using multiple DoH Servers ([I-D.pauly-dprive-oblivious-doh]). If this resolution fails, Privacy-Sensitive Connections will fail. All other connections will use the last resort, the default Direct Resolvers.

5. The default Direct Resolver, generally the resolver provisioned by the local router, is used as the last resort for any connection that is not explicitly Privacy-Sensitive [RFC2132] [RFC8106].

If the system allows the user to specify a preferred encrypted resolver, such as allowing the user to manually configure a DoH server URI to use by default, the use of this resolver SHOULD come between steps 2 and 3. This ensures that VPN-managed and locally-accessible names remain accessible while all other names are resolved using the user preference.

Resolution on behalf of system traffic, such as interactions required to detect and access a Captive Network Portal, require the use of the default Direct Resolver. System traffic SHOULD have an exception to this algorithm, and only use Steps 2 and 5 (those that use a resolver provisioned by the local network). Further deployment considerations for captive networks and walled-garden networks can be found in Section 6.2.3.

3.4. Oblivious Resolution

For all privacy-sensitive connection queries for names that do not correspond to a Designated DoH Server, the client SHOULD use Oblivious DoH to help conceal its IP address from eavesdroppers and untrusted resolvers.

Disassociation of client IPs from query content is achieved by using Oblivious DoH [I-D.pauly-dprive-oblivious-doh]. This extension to DoH allows a client to encrypt a query with a target DoH server's public key, and proxy the query through another server. The query is packaged with a unique client-defined symmetric key that is used to sign the DNS answer, which is sent back to the client via the proxy.

All DoH Servers that are used as Designated DoH Servers by the client MUST support being both an Oblivious Proxy and an Oblivious Target, as described in the server requirements (Section 4).

Since each Designated DoH Server can act as one of two roles in an proxied exchange, there are $(N) * (N - 1) / 2$ possible pairs of servers, where N is the number of whitelisted servers. While clients SHOULD use a variety of server pairs in rotation to decrease the ability for any given server to track client queries, it is not expected that all possible combinations will be used. Some combinations will be able to handle more load than others, and some will have better latency properties than others. To optimize performance, clients SHOULD maintain statistics to track the performance characteristics and success rates of particular pairs.

Clients that are performing Oblivious DoH resolution SHOULD fall back to another pair of servers if a first query times out, with a locally-determined limit for the number of fallback attempts that will be performed.

3.5. Handling Network Changes

Whenever a client joins a new network, it SHOULD wait to receive local configuration for resolvers before using any Designated DoH servers. The local network might be authoritative for some names, or might require filtering.

Once the local configuration of the new network has been received, the client MAY use Designated DoH configuration that it discovered when associated with another network. These configurations can still be considered valid since they came from DNSSEC-signed records. However, it is possible that different resolver IP addresses would be returned when looking up the designated server on the new network, which can provide a more optimal route through the Internet, so clients SHOULD perform new queries to refresh their mappings by making queries on connection on this new interface.

4. Server Requirements

Any server deployment that provides a set of services within one or more domains, such as a CDN, can run a server node that allows clients to run Adaptive DNS. A new server node can be added at any time, and can be used once it is advertised to clients and can be validated and whitelisted. The system overall is intended to scale and provide improved performance as more nodes become available.

The basic requirements to participate as a server node in this architecture are described below.

4.1. Provide a DoH Server

Each server node is primarily defined by a DoH server [RFC8484] that is designated for a set of domains, and also provides Oblivious DoH functionality. As such, the DoH servers MUST be able to act as recursive resolvers that accept queries for records and domains beyond those for which the servers are specifically designated.

4.1.1. Oblivious DoH Proxy

The DoH servers MUST be able to act as Oblivious Proxies. In this function, they will proxy encrypted queries and answers between clients and Oblivious Target DoH servers.

4.1.2. Oblivious DoH Target

The DoH servers MUST be able to act as Oblivious Targets. In this function, they will accept encrypted proxied queries from clients via Oblivious Proxy DoH servers, and provide encrypted answers using client keys.

4.1.3. Keying Material

In order to support acting as an Oblivious Target, a DoH server needs to provide a public HPKE [I-D.irtf-cfrg-hpke] key that can be used to encrypt client queries. This key is advertised in the SVCB record [I-D.pauly-dprive-oblivious-doh].

DoH servers also SHOULD provide an ESNI [I-D.ietf-tls-esni] key to encrypt the Server Name Indication field in TLS handshakes to the DoH server.

4.2. Advertise the DoH Server

The primary mechanism for advertising a Designated DoH Server is a SVCB DNS record (Section 3.1). This record MUST contain both the URI Template of the DoH Server as well as the Oblivious DoH Public Key. It MAY contain the ESNI key [I-D.ietf-tls-esni].

Servers MUST ensure that any SVCB records are signed with DNSSEC [RFC4033].

4.3. Provide Extended Configuration as a Web PvD

Beyond providing basic DoH server functionality, server nodes SHOULD provide a mechanism that allows clients to look up properties and configuration for the server deployment. Amongst other information, this configuration can optionally contain a list of some popular domains for which this server is designated. Clients can use this list to optimize lookups for common names.

This set of extended configuration information is referred to as a Web Provisioning Domain, or a Web PvD. Provisioning Domains are sets of consistent information that clients can use to access networks, including rules for resolution and proxying. Generally, these PvDs are provisioned directly, such as by a local router or a VPN. [I-D.ietf-intarea-provisioning-domains] defines an extensible configuration dictionary that can be used to add information to local PvD configurations. Web PvDs share the same JSON configuration format, and share the registry of keys defined as "Additional Information PvD Keys".

If present, the PvD JSON configuration MUST be made available to clients that request the "application/pvd+json" media type in a GET request to the DoH server's URI [I-D.ietf-intarea-provisioning-domains]. Clients MUST include this media type as an Accept header in their GET requests, and servers MUST mark this media type as their Content-Type header in responses. If the PvD JSON format is not supported, the server MUST reply with HTTP status code 415 [RFC7231].

The "identifier" key in the JSON configuration SHOULD be the hostname of the DoH Server itself.

For Web PvDs, the "prefixes" key within the JSON configuration SHOULD contain an empty array.

The key "dnsZones", which contains an array of domains as strings [I-D.ietf-intarea-provisioning-domains], indicates the zones that belong to the PvD. Any zone that is listed in this array for a Web PvD MUST have a corresponding SVCB record that defines the DoH server as designated for the zone. Servers SHOULD include in this array any names that are considered default or well-known for the deployment, but is not required or expected to list all zones or domains for which it is designated. The trade-off here is that zones that are listed can be fetched and validated automatically by clients, thus removing a bootstrapping step in discovering mappings from domains to Designated DoH Servers.

Clients that retrieve the Web PvD JSON dictionary SHOULD perform an SVCB record query for each of the entries in the "dnsZones" array in order to populate the mappings of domains. These MAY be performed in an oblivious fashion, but MAY also be queried directly on the DoH server (since the information is not user-specific, but in response to generic server-driven content). Servers can choose to preemptively transfer the relevant SVCB records if the PvD information retrieval is done with an HTTP version that supports PUSH semantics. This allows the server to avoid a round trip in zone validation even before the client has started requested SVCB records. Once the client requests an SVCB record for one of the names included in the "dnsZones" array, the server can also include the SVCB records for the other names in the array in the Additional section of the DNS response.

This document also registers one new key in the Additional Information PvD Keys registry, to identify the URI Template for the DoH server Section 10. When included in Web PvDs, this URI MUST match the template in the SVCB DNS Record.

Beyond providing resolution configuration, the Web PvD configuration can be extended to offer information about proxies and other services offered by the server deployment. Such keys are not defined in this document.

5. Server Deployment Considerations

When servers designate DoH servers for their names, the specific deployment model can impact the effective privacy and performance characteristics.

5.1. Single Content Provider

If a name always resolves to server IP addresses that are hosted by a single content provider, the name ought to designate a single DoH server. This DoH server will be most optimal when it is designated by many or all names that are hosted by the same content provider. This ensures that clients can increase connection reuse to reduce latency in connection setup.

A DoH server that corresponds to the content provider that hosts content has an opportunity to tune the responses provided to a client based on the location inferred by the client IP address.

5.2. Multiple Content Providers

Some hostnames may resolve to server IP addresses that are hosted by multiple content providers. In such scenarios, the deployment may want to be able to control the percentage of traffic that flows to each content provider.

In these scenarios, there can either be:

- o multiple designated DoH servers that are advertised via SVCB DNS Records; or,
- o a single designated DoH server that can be referenced by one or more SVCB DNS Records, operated by a party that is aware of both content providers and can manage splitting the traffic.

If a server deployment wants to easily control the split of traffic between different content providers, it ought to use the latter model of using a single designated DoH server that can better control which IP addresses are provided to clients. Otherwise, if a client is aware of multiple DoH servers, it might use a single resolver exclusively, which may lead to inconsistent behavior between clients that choose different resolvers.

5.3. Avoid Narrow Deployments

Using designated DoH servers can improve the privacy of name resolution whenever a DoH server is designated by many different names within one or more domains. This limits the amount of information leaked to an attacker observing traffic between a client and a DoH server: the attacker only learns that the client might be resolving one of the many names for which the server is designated (or might be performing an Oblivious query).

However, if a deployment designates a given DoH server for only one name, or a very small set of names, then it becomes easier for an attacker to infer that a specific name is being accessed by a client. For this reason, deployments are encouraged to avoid deploying a DoH server that is only designated by a small number of names. Clients can also choose to only whitelist DoH servers that are associated with many names.

Beyond the benefits to privacy, having a larger number of names designate a given DoH server improves the opportunity for DoH connection reuse, which can improve the performance of name resolutions.

6. Local Resolver Deployment Considerations

A key goal of Adaptive DNS is that clients will be able to use Designated DoH Servers to improve the privacy of queries, without entirely bypassing local network authority and policy. For example, if a client is attached to an enterprise Wi-Fi network that provides access and resolution for private names not generally accessible on the Internet, such names will only be usable when a local resolver is used.

In order to achieve this, a local network can advertise itself as authoritative for a domain, allowing it to be used prior to external servers in the client resolution algorithm Section 3.3.

6.1. Designating Local DoH Servers

If a local network wants to have clients send queries for a set of private domains to its own resolver, it needs to define an explicit provisioning domain [I-D.ietf-intarea-provisioning-domains]. The PvD RA option SHOULD set the H-flag to indicate that Additional Information is available. This Additional Information JSON object SHOULD include both the "dohTemplate" and "dnsZones" keys to define the local DoH server and the domains over which it claims authority.

In order to validate that a local resolver is designated for a given zone, the client SHOULD issue a SVCB record query for the names specified in the PvD information, using the DoH server specified in the PvD information. If there is no SVCB record for a name that points to the DoH server that can be validated using DNSSEC, the client SHOULD NOT automatically create a designation from the domain name to DoH server. See specific use cases in Section 6.2 for cases in which a local resolver may still be used.

Although local Designated DoH Servers MAY support proxying Oblivious DoH queries, a client SHOULD NOT select one of these servers as an Oblivious Proxy. Doing so might reveal the client's location to the Target based on the address of the proxy, which could contribute to deanonymizing the client. Clients can make an exception to this behavior if the DoH server designated by the local network is known to be a non-local service, such as when a local network configures a centralized public resolver to handle its DNS operations.

6.2. Local Use Cases

The various use cases for selecting locally-provisioned resolvers require different approaches for deployment and client resolution. The following list is not exhaustive, but provides guidance on how these scenarios can be achieved using the Adaptive DNS algorithm.

6.2.1. Accessing Local-Only Resolvable Content

Some names are not resolvable using generic DNS resolvers, but require using a DNS server that can resolve private names. This is common in enterprise scenarios, in which an enterprise can have a set of private names that it allows to be resolved when connected to a VPN or an enterprise-managed Wi-Fi network. In this case, clients that do not use the locally-provisioned resolver will fail to resolve private names.

In these scenarios, the local network SHOULD designate a local DoH server for the domains that are locally resolvable. For example, an enterprise that owns "private.example.org" would advertise "private.example.org" in its PvD information along with a DoH URI template. Clients could then use that locally-configured resolver with names under "private.example.org" according to the rules in Section 6.1.

In general, clients SHOULD only create designated DoH server associations when they can validate a SVCB record using DNSSEC. However, some deployments of private names might not want to sign all private names within a zone. There are thus a few possible deployment models:

- o "private.example.org" does have a DNSSEC-signed SVCB record that points to the local DoH server. The client requests the SVCB record for "private.example.org" using the local DoH server that is specified in the PvD information, and from that point on uses the local DoH server for names under "private.example.org".
- o Instead of signing "private.example.org", the deployment provides a DNSSEC-signed SVCB record for "example.org", thus steering all resolution under "example.org" to the local resolver.
- o No DNSSEC-signed SVCB record designates the local server. In this case, clients have a hint that the local network can serve names under "private.example.org", but do not have a way to validate the designation. Clients can in this case try to resolve names using external servers (such as via Oblivious DoH), and then MAY fall back to using locally-provisioned resolvers if the names do not resolve externally. This approach has the risk of exposing private names to public resolvers, which can be undesirable for certain enterprise deployments. Alternatively, if the client trusts the local network based on specific policy configured on the client, it can choose to resolve these names locally first. Note that this approach risks exposing names to a potentially malicious network that is masquerading as an authority for private names if the network cannot be validated in some other manner.

Deployments SHOULD use the one of the first two approaches (signing their records) whenever possible; the case of providing unsigned names is only described as a possibility for handling legacy enterprise deployments. Clients SHOULD choose to ignore any locally designated names that are not signed unless there is a specific policy configuration on the client.

6.2.2. Accessing Locally Optimized Content

Other names may be resolvable both publicly and on the local resolver, but have more optimized servers that are accessible only via the local network. For example, a Wi-Fi provider may provide access to a cache of video content that provides lower latency than publicly-accessible caches.

Names that are hosted locally in this way SHOULD use a designation with a DNSSEC-signed SVCB record for the name. If a client discovers that a local resolver is designated for a given name, the client SHOULD prefer using connections to this locally-hosted content rather than names resolved externally.

Note that having a DNSSEC-signed designation to the local resolver provides a clear indication that the entity that manages a given name has an explicit relationship with the local network provider.

6.2.3. Walled-Garden and Captive Network Deployments

Some networks do not provide any access to the general Internet, but host local content that clients can access. For example, a network on an airplane can give access to flight information and in-flight media, but will not allow access to any external hosts or DNS servers. These networks are often described as "walled-gardens".

Captive networks [I-D.ietf-capport-architecture] are similar in that they block access to external hosts, although they can provide generic access after some time.

If a walled-garden or captive network defines a PvD with additional information, it can define zones for names that it hosts, such as "airplane.example.com". It can also provide a locally-hosted encrypted DNS server.

However, if such a network does not support explicitly advertising local names, clients that try to establish connections to DoH servers will experience connection failures. In these cases, system traffic that is used for connecting to captive portals SHOULD use local resolvers. In addition, clients MAY choose to fall back to using direct resolution without any encryption if they determine that all connectivity is blocked otherwise. Note that this comes with a risk of a network blocking connections in order to induce this fallback behavior, so clients might want to inform users about this possible attack where appropriate, or prefer to not fall back if there is a concern about leaking user data.

6.2.4. Network-Based Filtering

Some networks currently rely on manipulating DNS name resolution in order to apply content filtering rules to clients associated with the network. Using encrypted DNS resolvers that are not participating in this filtering can bypass such enforcement. However, simply blocking connections for filtering is indistinguishable from a malicious attack from a client's perspective.

In order to indicate the presence of filtering requirements, a network deployment can add the "requiredDNSFiltering" and "dnsFilteredZones" keys to its PvD information. The "dnsFilteredZones" entry can contain an array of strings, each of which is a domain name that the network requires clients to resolve using the local resolver. If the array contains the string ".", it

indicates the network requires filtering for all domains. If "requiredDNSFiltering" is present with a boolean value of true, the network is indicating that it expects all client systems to send the names indicated by "dnsFilteredZones" to the local resolver. If "requiredDNSFiltering" is not present or set to false, then the filtering service is considered to be optional for clients that want to use it as a service to enforce desired policy.

Clients that receive indication of filtering requirements SHOULD NOT use any other resolver for the filtered domains, but treat the network as claiming authority. However, since this filtering cannot be authenticated, this behavior SHOULD NOT be done silently without user consent.

Networks that try to interfere with connections to encrypted DNS resolvers without indicating a requirement for filtering cannot be distinguished from misconfigurations or network attacks. Clients MAY choose to avoid sending any user-initiated connections on such networks to prevent malicious interception.

7. Performance Considerations

One of the challenges of using non-local DNS servers (such as cloud-based DoH servers) is that recursive queries made by these servers will originate from an IP address that is not necessarily geographically related to the client. Many DNS servers make assumptions about the geographic locality of clients to their recursive resolvers to optimize answers. To avoid this problem, the client's subnet can be forwarded to the authoritative server by the recursive using the EDNS0 Client Subnet feature. Oblivious DoH discourages this practice for privacy reasons. However, sharing this subnet, while detrimental to privacy, can result in better targeted DNS resolutions.

Adaptive DNS splits DoH queries into two sets: those made to Designated DoH Servers, and those made to Oblivious DoH servers. Oblivious queries are sensitive for privacy, and can encounter performance degradation as a result of not using the client subnet. Queries to designated DoH servers, on the other hand, are sent directly by clients, so the client IP address is made available to these servers. Since these servers are designated by the authority for the names, they can use the IP address subnet information to tune DNS answers.

Based on these properties, clients SHOULD prefer lookups via Designated DoH Servers over oblivious mechanisms whenever possible. Servers can encourage this by setting large TTLs for SVCB records and using longer TTLs for responses returned by their Designated DoH

Server endpoints which can be more confident they have accurate addressing information.

8. Security Considerations

In order to avoid interception and modification of the information retrieved by clients using Adaptive DNS, all exchanges between clients and servers are performed over encrypted connections, e.g., TLS.

Malicious adversaries may block client connections to all DoH or Oblivious DoH services as a Denial-of-Service (DoS) measure. Clients which cannot connect to any proxy may, by local policy, fall back to unencrypted DNS if this occurs.

9. Privacy Considerations

Clients must be careful in determining to which DoH servers they send queries directly without proxying. A malicious DoH server that can direct queries to itself can track or profile client activity. In order to avoid the possibility of a spoofed SVCB record designating a malicious DoH server for a name, clients MUST ensure that such records validate using DNSSEC [RFC4033].

Even servers that are officially designated can risk leaking or logging information about client lookups. Such risk can be mitigated by further restricting the list of DoH servers that are whitelisted for direct use based on client policy.

Using Oblivious DoH reduces the risk that a single DoH server can track or profile a client. However, clients should exercise caution when using Oblivious DoH responses from resolvers that do not carry DNSSEC signatures. An adversarial Oblivious Target resolver that wishes to learn the IP address of clients requesting resolution for sensitive domains can redirect clients to unique addresses of its choosing. Clients that use these answers when establishing TLS connections may then leak their local IP address to chosen server. Thus, when Oblivious DoH answers are returned without DNSSEC, Privacy-Sensitive Connections concerned about this attack SHOULD conceal their IP address via a TLS- or HTTP-layer proxy or some other tunneling mechanism.

An adversary able to see traffic on each path segment of a DoH or Oblivious DoH query (e.g., from client to proxy, proxy to target, and target to an authoritative DNS server) can link queries to specific clients with high probability. Failure to observe traffic on any one of these path segments makes this linkability increasingly difficult. For example, if an adversary can only observe traffic between a

client and proxy and egress traffic from a target, then it may be difficult to identify a specific client's query among the recursive queries generated by the target.

10. IANA Considerations

10.1. DoH Template PvD Key

This document adds a key to the "Additional Information PvD Keys" registry [I-D.ietf-intarea-provisioning-domains].

JSON key	Description	Type	Example
dohTemplate	DoH URI Template [RFC8484]	String	"https://dnsserver.example.net/dns-query{?dns}"

10.2. DNS Filtering PvD Keys

This document adds a key to the "Additional Information PvD Keys" registry [I-D.ietf-intarea-provisioning-domains].

JSON key	Description	Type	Example
requireDNSFiltering	A flag to indicate that the network requires filtering all DNS traffic using the provisioned resolver.	Boolean	true
dnsFilteredZones	A list of DNS domains as strings that represent domains that can be filtered by the provisioned resolver.	Array of Strings	["."]

Any network that sets the "requireDNSFiltering" boolean to false but provides "dnsFilteredZones" advertises the optional service of filtering on the provisioned network.

An "." in the "dnsFilteredZones" array represents a wildcard, which can be used to indicate that the network is requesting to filter all

names. Any more specific string represents a domain that requires filtering on the network.

10.3. DoH URI Template DNS Parameter

If present, this parameters indicates the URI template of a DoH server that is designated for use with the name being resolved. This is a string encoded as UTF-8 characters.

Name: dohuri

SvcParamKey: TBD

Meaning: URI template for a designated DoH server

Reference: This document.

11. Acknowledgments

Thanks to Erik Nygren, Lorenzo Colitti, Tommy Jensen, Mikael Abrahamsson, Ben Schwartz, Ask Hansen, Leif Hedstrom, Tim McCoy, Stuart Cheshire, Miguel Vega, Joey Deng, Ted Lemon, and Elliot Briggs for their feedback and input on this document.

12. References

12.1. Normative References

- [I-D.ietf-intarea-provisioning-domains]
Pfister, P., Vyncke, E., Pauly, T., Schinazi, D., and W. Shao, "Discovering Provisioning Domain Names and Data", draft-ietf-intarea-provisioning-domains-08 (work in progress), October 2019.
- [I-D.ietf-tls-esni]
Rescorla, E., Oku, K., Sullivan, N., and C. Wood, "Encrypted Server Name Indication for TLS 1.3", draft-ietf-tls-esni-04 (work in progress), July 2019.
- [I-D.irtf-cfrg-hpke]
Barnes, R. and K. Bhargavan, "Hybrid Public Key Encryption", draft-irtf-cfrg-hpke-00 (work in progress), July 2019.

- [I-D.nygren-dnsop-svcb-httpssvc]
Schwartz, B., Bishop, M., and E. Nygren, "Service binding and parameter specification via the DNS (DNS SVCB and HTTPSSVC)", draft-nygren-dnsop-svcb-httpssvc-00 (work in progress), September 2019.
- [I-D.pauly-dprive-oblivious-doh]
Kinnear, E., Pauly, T., Wood, C., and P. McManus, "Oblivious DNS Over HTTPS", draft-pauly-dprive-oblivious-doh-00 (work in progress), October 2019.
- [RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, DOI 10.17487/RFC4033, March 2005, <<https://www.rfc-editor.org/info/rfc4033>>.
- [RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <<https://www.rfc-editor.org/info/rfc7231>>.
- [RFC7858] Hu, Z., Zhu, L., Heidemann, J., Mankin, A., Wessels, D., and P. Hoffman, "Specification for DNS over Transport Layer Security (TLS)", RFC 7858, DOI 10.17487/RFC7858, May 2016, <<https://www.rfc-editor.org/info/rfc7858>>.
- [RFC8484] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <<https://www.rfc-editor.org/info/rfc8484>>.

12.2. Informative References

- [I-D.ietf-capport-architecture]
Larose, K. and D. Dolson, "CAPPORT Architecture", draft-ietf-capport-architecture-04 (work in progress), June 2019.
- [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>>.
- [RFC2132] Alexander, S. and R. Droms, "DHCP Options and BOOTP Vendor Extensions", RFC 2132, DOI 10.17487/RFC2132, March 1997, <<https://www.rfc-editor.org/info/rfc2132>>.

- [RFC8106] Jeong, J., Park, S., Beloeil, L., and S. Madanapalli,
"IPv6 Router Advertisement Options for DNS Configuration",
RFC 8106, DOI 10.17487/RFC8106, March 2017,
<<https://www.rfc-editor.org/info/rfc8106>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC
2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174,
May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

Authors' Addresses

Eric Kinnear
Apple Inc.
One Apple Park Way
Cupertino, California 95014
United States of America

Email: ekinnear@apple.com

Tommy Pauly
Apple Inc.
One Apple Park Way
Cupertino, California 95014
United States of America

Email: tpauly@apple.com

Chris Wood
Apple Inc.
One Apple Park Way
Cupertino, California 95014
United States of America

Email: cawood@apple.com

Patrick McManus
Fastly

Email: mcmanus@ducksong.com

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: 21 August 2022

E. Kinnear
Apple Inc.
P. McManus
Fastly
T. Pauly
Apple Inc.
T. Verma
C.A. Wood
Cloudflare
17 February 2022

Oblivious DNS Over HTTPS
draft-pauly-dprive-oblivious-doh-11

Abstract

This document describes a protocol that allows clients to hide their IP addresses from DNS resolvers via proxying encrypted DNS over HTTPS (DoH) messages. This improves privacy of DNS operations by not allowing any one server entity to be aware of both the client IP address and the content of DNS queries and answers.

This experimental protocol is developed outside the IETF and is published here to guide implementation, ensure interoperability among implementations, and enable wide-scale experimentation.

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 <https://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 21 August 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction	2
1.1. Specification of Requirements	3
2. Terminology	3
3. Deployment Requirements	4
4. HTTP Exchange	4
4.1. HTTP Request	5
4.2. HTTP Request Example	6
4.3. HTTP Response	6
4.4. HTTP Response Example	7
4.5. HTTP Metadata	8
5. Configuration and Public Key Format	8
6. Protocol Encoding	9
6.1. Message Format	9
6.2. Encryption and Decryption Routines	11
7. Oblivious Client Behavior	12
8. Oblivious Target Behavior	13
9. Compliance Requirements	14
10. Experiment Overview	14
11. Security Considerations	14
11.1. Denial of Service	16
11.2. Proxy Policies	16
11.3. Authentication	16
12. IANA Considerations	17
12.1. Oblivious DoH Message Media Type	17
13. Acknowledgments	18
14. References	18
14.1. Normative References	18
14.2. Informative References	19
Appendix A. Use of Generic Proxy Services	20
Authors' Addresses	20

1. Introduction

DNS Over HTTPS (DoH) [RFC8484] defines a mechanism to allow DNS messages to be transmitted in HTTP messages protected with TLS. This provides improved confidentiality and authentication for DNS interactions in various circumstances.

While DoH can prevent eavesdroppers from directly reading the contents of DNS exchanges, clients cannot send DNS queries to and receive answers from servers without revealing their local IP address (and thus information about the identity or location of the client) to the server.

Proposals such as Oblivious DNS ([I-D.annex-dprive-oblivious-dns]) increase privacy by ensuring no single DNS server is aware of both the client IP address and the message contents.

This document defines Oblivious DoH, an experimental protocol built on DoH that permits proxied resolution, in which DNS messages are encrypted so that no server can independently read both the client IP address and the DNS message contents.

As with DoH, DNS messages exchanged over Oblivious DoH are fully-formed DNS messages. Clients that want to receive answers that are relevant to the network they are on without revealing their exact IP address can thus use the EDNS0 Client Subnet option [RFC7871], Section 7.1.2 to provide a hint to the resolver using Oblivious DoH.

This mechanism is intended to be used as one mechanism for resolving privacy-sensitive content in the broader context of DNS privacy.

This experimental protocol is developed outside the IETF and is published here to guide implementation, ensure interoperability among implementations, and enable wide-scale experimentation. See Section 10 for more details about the experiment.

1.1. Specification of Requirements

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Terminology

This document defines the following terms:

Oblivious Proxy: An HTTP server that proxies encrypted DNS queries and responses between a Client and an Oblivious Target, and is identified by a URI template [RFC6570] (see Section 4.1). Note that this Oblivious Proxy is not acting as a full HTTP proxy, but is instead a specialized server used to forward oblivious DNS messages.

Oblivious Target: An HTTP server that receives and decrypts encrypted Client DNS queries from an Oblivious Proxy, and returns encrypted DNS responses via that same Proxy. In order to provide DNS responses, the Target can be a DNS resolver, be co-located with a resolver, or forward to a resolver.

Throughout the rest of this document, we use the terms Proxy and Target to refer to an Oblivious Proxy and Oblivious Target, respectively.

3. Deployment Requirements

Oblivious DoH requires, at a minimum:

- * An Oblivious Proxy server, identified by a URI template.
- * An Oblivious Target server. The Target and Proxy are expected to be non-colluding (see Section 11).
- * One or more Target public keys for encrypting DNS queries send to a Target via a Proxy (Section 5). These keys guarantee that only the intended Target can decrypt Client queries.

The mechanism for discovering and provisioning the Proxy URI template and Target public keys is out of scope of this document.

4. HTTP Exchange

Unlike direct resolution, oblivious hostname resolution over DoH involves three parties:

1. The Client, which generates queries.
2. The Proxy, which receives encrypted queries from the Client and passes them on to a Target.
3. The Target, which receives proxied queries from the Client via the Proxy and produces proxied answers.

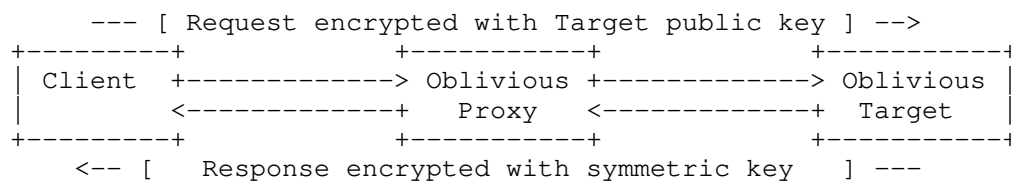


Figure 1: Oblivious DoH Exchange

4.1. HTTP Request

Oblivious DoH queries are created by the Client, and sent to the Proxy as HTTP requests using the POST method. Clients are configured with a Proxy URI Template [RFC6570] and the Target URI. The scheme for both the Proxy URI Template and the Target URI MUST be "https". The Proxy URI Template uses the Level 3 encoding defined in Section 1.2 of [RFC6570], and contains two variables: "targethost", which indicates the host name of the Target server; and "targetpath", which indicates the path on which the Target is accessible. Examples of Proxy URI Templates are shown below:

```
https://dnsproxy.example/dns-query{?targethost,targetpath}  
https://dnsproxy.example/{targethost}/{targetpath}
```

The URI Template MUST contain both the "targethost" and "targetpath" variables exactly once, and MUST NOT contain any other variables. The variables MUST be within the path or query components of the URI. Clients MUST ignore configurations that do not conform to this template. See Section 4.2 for an example request.

Oblivious DoH messages have no cache value since both requests and responses are encrypted using ephemeral key material. Requests and responses MUST NOT be cached.

Clients MUST set the HTTP Content-Type header to "application/oblivious-dns-message" to indicate that this request is an Oblivious DoH query intended for proxying. Clients also SHOULD set this same value for the HTTP Accept header.

A correctly encoded request has the HTTP Content-Type header "application/oblivious-dns-message", uses the HTTP POST method, and contains "targethost" and "targetpath" variables. If the Proxy fails to match the "targethost" and "targetpath" variables from the path, it MUST treat the request as malformed. The Proxy constructs the URI of the Target with the "https" scheme, using the value of "targethost" as the URI host, and the percent-decoded value of "targetpath" as the URI path. Proxies MUST check that Client requests are correctly encoded, and MUST return a 4xx (Client Error) if the check fails, along with the Proxy-Status response header with an "error" parameter of type "http_request_error" [I-D.ietf-httpbis-proxy-status].

Proxies MAY choose to not forward connections to non-standard ports. In such cases, Proxies can indicate the error with a 403 response status code, along with a Proxy-Status response header with an "error" parameter of type "http_request_denied", along with an appropriate explanation in "details".

If the Proxy cannot establish a connection to the Target, it can indicate the error with a 502 response status code, along with a Proxy-Status response header with an "error" parameter whose type indicates the reason. For example, if DNS resolution fails, the error type might be "dns_timeout", whereas if the TLS connection failed the error type might be "tls_protocol_error".

Upon receipt of requests from a Proxy, Targets MUST validate that the request has the HTTP Content-Type header "application/oblivious-dns-message" and uses the HTTP POST method. Targets can respond with a 4xx response status code if this check fails.

4.2. HTTP Request Example

The following example shows how a Client requests that a Proxy, "dnspoxy.example", forwards an encrypted message to "dnstarget.example". The URI Template for the Proxy is "https://dnspoxy.example/dns-query{?targethost,targetpath}". The URI for the Target is "https://dnstarget.example/dns-query".

```
:method = POST
:scheme = https
:authority = dnspoxy.example
:path = /dns-query?targethost=dnstarget.example&targetpath=/dns-query
accept = application/oblivious-dns-message
content-type = application/oblivious-dns-message
content-length = 106
```

<Bytes containing an encrypted Oblivious DNS query>

The Proxy then sends the following request on to the Target:

```
:method = POST
:scheme = https
:authority = dnstarget.example
:path = /dns-query
accept = application/oblivious-dns-message
content-type = application/oblivious-dns-message
content-length = 106
```

<Bytes containing an encrypted Oblivious DNS query>

4.3. HTTP Response

The response to an Oblivious DoH query is generated by the Target. It MUST set the Content-Type HTTP header to "application/oblivious-dns-message" for all successful responses. The body of the response contains an encrypted DNS message; see Section 6.

The response from a Target MUST set the Content-Type HTTP header to "application/oblivious-dns-message" which MUST be forwarded by the Proxy to the Client. A Client MUST only consider a response which contains the Content-Type header in the response before processing the payload. A response without the appropriate header MUST be treated as an error and be handled appropriately. All other aspects of the HTTP response and error handling are inherited from standard DoH.

Proxies forward responses from the Target to client, without any modifications to the body or status code. The Proxy also SHOULD add a Proxy-Status response header with an "received-status" parameter indicating that the status code was generated by the Target.

Note that if a Client receives a 3xx status code and chooses to follow a redirect, the subsequent request MUST also be performed through a Proxy in order to avoid directly exposing requests to the Target.

Requests that cannot be processed by the Target result in 4xx (Client Error) responses. If the Target and Client keys do not match, it is an authorization failure (HTTP status code 401; see Section 3.1 of [RFC7235]). Otherwise, if the Client's request is invalid, such as in the case of decryption failure, wrong message type, or deserialization failure, this is a bad request (HTTP status code 400; see Section 6.5.1 of [RFC7231]).

Even in case of DNS responses indicating failure, such as SERVFAIL or NXDOMAIN, a successful HTTP response with a 2xx status code is used as long as the DNS response is valid. This is identical to how DoH [RFC8484] handles HTTP response codes.

4.4. HTTP Response Example

The following example shows a 2xx (Successful) response that can be sent from a Target to a Client via a Proxy.

```
:status = 200
content-type = application/oblivious-dns-message
content-length = 154

<Bytes containing an encrypted Oblivious DNS response>
```

4.5. HTTP Metadata

Proxies forward requests and responses between Clients and Targets as specified in Section 4.1. Metadata sent with these messages could inadvertently weaken or remove Oblivious DoH privacy properties. Proxies **MUST NOT** send any Client-identifying information about Clients to Targets, such as "Forwarded" HTTP headers [RFC7239]. Additionally, Clients **MUST NOT** include any private state in requests to Proxies, such as HTTP cookies. See Section 11.3 for related discussion about Client authentication information.

5. Configuration and Public Key Format

In order send a message to a Target, the Client needs to know a public key to use for encrypting its queries. The mechanism for discovering this configuration is out of scope of this document.

Servers ought to rotate public keys regularly. It is **RECOMMENDED** that servers rotate keys every day. Shorter rotation windows reduce the anonymity set of Clients that might use the public key, whereas longer rotation windows widen the timeframe of possible compromise.

An Oblivious DNS public key configuration is a structure encoded, using TLS-style encoding [RFC8446], as follows:

```
struct {
    uint16 kem_id;
    uint16 kdf_id;
    uint16 aead_id;
    opaque public_key<1..2^16-1>;
} ObliviousDoHConfigContents;

struct {
    uint16 version;
    uint16 length;
    select (ObliviousDoHConfig.version) {
        case 0x0001: ObliviousDoHConfigContents contents;
    }
} ObliviousDoHConfig;

ObliviousDoHConfig ObliviousDoHConfigs<1..2^16-1>;
```

The ObliviousDoHConfigs structure contains one or more ObliviousDoHConfig structures in decreasing order of preference. This allows a server to support multiple versions of Oblivious DoH and multiple sets of Oblivious DoH parameters.

An `ObliviousDoHConfig` contains a versioned representation of an Oblivious DoH configuration, with the following fields.

version The version of Oblivious DoH for which this configuration is used. Clients MUST ignore any `ObliviousDoHConfig` structure with a version they do not support. The version of Oblivious DoH specified in this document is 0x0001.

length The length, in bytes, of the next field.

contents An opaque byte string whose contents depend on the version. For this specification, the contents are an `ObliviousDoHConfigContents` structure.

An `ObliviousDoHConfigContents` contains the information needed to encrypt a message under `ObliviousDoHConfigContents.public_key` such that only the owner of the corresponding private key can decrypt the message. The values for `ObliviousDoHConfigContents.kem_id`, `ObliviousDoHConfigContents.kdf_id`, and `ObliviousDoHConfigContents.aead_id` are described in Section 7 of [HPKE]. The fields in this structure are as follows:

kem_id The HPKE KEM identifier corresponding to `public_key`. Clients MUST ignore any `ObliviousDoHConfig` structure with a key using a KEM they do not support.

kdf_id The HPKE KDF identifier corresponding to `public_key`. Clients MUST ignore any `ObliviousDoHConfig` structure with a key using a KDF they do not support.

aead_id The HPKE AEAD identifier corresponding to `public_key`. Clients MUST ignore any `ObliviousDoHConfig` structure with a key using an AEAD they do not support.

public_key The HPKE public key used by the Client to encrypt Oblivious DoH queries.

6. Protocol Encoding

This section includes encoding and wire format details for Oblivious DoH, as well as routines for encrypting and decrypting encoded values.

6.1. Message Format

There are two types of Oblivious DoH messages: Queries (0x01) and Responses (0x02). Both messages carry the following information:

1. A DNS message, which is either a Query or Response, depending on context.
2. Padding of arbitrary length which MUST contain all zeros.

They are encoded using the following structure:

```
struct {  
    opaque dns_message<1..2^16-1>;  
    opaque padding<0..2^16-1>;  
} ObliviousDoHMessagePlaintext;
```

Both Query and Response messages use the ObliviousDoHMessagePlaintext format.

```
ObliviousDoHMessagePlaintext ObliviousDoHQuery;  
ObliviousDoHMessagePlaintext ObliviousDoHResponse;
```

An encrypted ObliviousDoHMessagePlaintext is carried in a ObliviousDoHMessage message, encoded as follows:

```
struct {  
    uint8 message_type;  
    opaque key_id<0..2^16-1>;  
    opaque encrypted_message<1..2^16-1>;  
} ObliviousDoHMessage;
```

The ObliviousDoHMessage structure contains the following fields:

message_type A one-byte identifier for the type of message. Query messages use message_type 0x01, and Response messages use message_type 0x02.

key_id The identifier of the corresponding ObliviousDoHConfigContents key. This is computed as `Expand(Extract("", config), "odoh key id", Nh)`, where config is the ObliviousDoHConfigContents structure and Extract, Expand, and Nh are as specified by the HPKE cipher suite KDF corresponding to config.kdf_id.

encrypted_message An encrypted message for the Oblivious Target (for Query messages) or Client (for Response messages). Implementations MAY enforce limits on the size of this field depending on the size of plaintext DNS messages. (DNS queries, for example, will not reach the size limit of $2^{16}-1$ in practice.)

The contents of `ObliviousDoHMessage.encrypted_message` depend on `ObliviousDoHMessage.message_type`. In particular, `ObliviousDoHMessage.encrypted_message` is an encryption of a `ObliviousDoHQuery` if the message is a Query, and `ObliviousDoHResponse` if the message is a Response.

6.2. Encryption and Decryption Routines

Clients use the following utility functions for encrypting a Query and decrypting a Response as described in Section 7.

`encrypt_query_body`: Encrypt an Oblivious DoH query.

```
def encrypt_query_body(pkR, key_id, Q_plain):
    enc, context = SetupBaseS(pkR, "odoh query")
    aad = 0x01 || len(key_id) || key_id
    ct = context.Seal(aad, Q_plain)
    Q_encrypted = enc || ct
    return Q_encrypted
```

`decrypt_response_body`: Decrypt an Oblivious DoH response.

```
def decrypt_response_body(context, Q_plain, R_encrypted, resp_nonce):
    aead_key, aead_nonce = derive_secrets(context, Q_plain, resp_nonce)
    aad = 0x02 || len(resp_nonce) || resp_nonce
    R_plain, error = Open(key, nonce, aad, R_encrypted)
    return R_plain, error
```

The `derive_secrets` function is described below.

Targets use the following utility functions in processing queries and producing responses as described in Section 8.

`setup_query_context`: Set up an HPKE context used for decrypting an Oblivious DoH query.

```
def setup_query_context(skR, key_id, Q_encrypted):
    enc || ct = Q_encrypted
    context = SetupBaseR(enc, skR, "odoh query")
    return context
```

`decrypt_query_body`: Decrypt an Oblivious DoH query.

```
def decrypt_query_body(context, key_id, Q_encrypted):
    aad = 0x01 || len(key_id) || key_id
    enc || ct = Q_encrypted
    Q_plain, error = context.Open(aad, ct)
    return Q_plain, error
```

`derive_secrets`: Derive keying material used for encrypting an Oblivious DoH response.

```
def derive_secrets(context, Q_plain, resp_nonce):
    secret = context.Export("odoh response", Nk)
    salt = Q_plain || len(resp_nonce) || resp_nonce
    prk = Extract(salt, secret)
    key = Expand(odoh_prk, "odoh key", Nk)
    nonce = Expand(odoh_prk, "odoh nonce", Nn)
    return key, nonce
```

The `random(N)` function returns `N` cryptographically secure random bytes from a good source of entropy [RFC4086]. The `max(A, B)` function returns `A` if `A > B`, and `B` otherwise.

`encrypt_response_body`: Encrypt an Oblivious DoH response.

```
def encrypt_response_body(R_plain, aead_key, aead_nonce, resp_nonce):
    aad = 0x02 || len(resp_nonce) || resp_nonce
    R_encrypted = Seal(aead_key, aead_nonce, aad, R_plain)
    return R_encrypted
```

7. Oblivious Client Behavior

Let `M` be a DNS message (query) a Client wishes to protect with Oblivious DoH. When sending an Oblivious DoH Query for resolving `M` to an Oblivious Target with `ObliviousDoHConfigContents config`, a Client does the following:

1. Create an `ObliviousDoHQuery` structure, carrying the message `M` and padding, to produce `Q_plain`.
2. Deserialize `config.public_key` to produce a public key `pkR` of type `config.kem_id`.
3. Compute the encrypted message as `Q_encrypted = encrypt_query_body(pkR, key_id, Q_plain)`, where `key_id` is as computed in Section 6. Note also that `len(key_id)` outputs the length of `key_id` as a two-byte unsigned integer.
4. Output an `ObliviousDoHMessage` message `Q` where `Q.message_type = 0x01`, `Q.key_id` carries `key_id`, and `Q.encrypted_message = Q_encrypted`.

The Client then sends `Q` to the Proxy according to Section 4.1. Once the Client receives a response `R`, encrypted as specified in Section 8, it uses `decrypt_response_body` to decrypt `R.encrypted_message` (using `R.key_id` as a nonce) and produce `R_plain`. Clients MUST validate `R_plain.padding` (as all zeros) before using `R_plain.dns_message`.

8. Oblivious Target Behavior

Targets that receive a Query message `Q` decrypt and process it as follows:

1. Look up the `ObliviousDoHConfigContents` according to `Q.key_id`. If no such key exists, the Target MAY discard the query, and if so, it MUST return a 401 (Unauthorized) response to the Proxy. Otherwise, let `skR` be the private key corresponding to this public key, or one chosen for trial decryption.
2. Compute `context = setup_query_context(skR, Q.key_id, Q.encrypted_message)`.
3. Compute `Q_plain, error = decrypt_query_body(context, Q.key_id, Q.encrypted_message)`.
4. If no error was returned, and `Q_plain.padding` is valid (all zeros), resolve `Q_plain.dns_message` as needed, yielding a DNS message `M`. Otherwise, if an error was returned or the padding was invalid, return a 400 (Client Error) response to the Proxy.
5. Create an `ObliviousDoHResponseBody` structure, carrying the message `M` and padding, to produce `R_plain`.
6. Create a fresh nonce `resp_nonce = random(max(Nn, Nk))`.
7. Compute `aead_key, aead_nonce = derive_secrets(context, Q_plain, resp_nonce)`.
8. Compute `R_encrypted = encrypt_response_body(R_plain, aead_key, aead_nonce, resp_nonce)`. The `key_id` field used for encryption carries `resp_nonce` in order for Clients to derive the same secrets. Also, the Seal function is that which is associated with the HPKE AEAD.
9. Output an `ObliviousDoHMessage` message `R` where `R.message_type = 0x02`, `R.key_id = resp_nonce`, and `R.encrypted_message = R_encrypted`.

The Target then sends R in a 2xx (Successful) response to the Proxy; see Section 4.3. The Proxy forwards the message R without modification back to the Client as the HTTP response to the Client's original HTTP request. In the event of an error (non 2xx status code), the Proxy forwards the Target error to the Client; see Section 4.3.

9. Compliance Requirements

Oblivious DoH uses HPKE for public key encryption [HPKE]. In the absence of an application profile standard specifying otherwise, a compliant Oblivious DoH implementation MUST support the following HPKE cipher suite:

- * KEM: DHKEM(X25519, HKDF-SHA256) (see [HPKE], Section 7.1)
- * KDF: HKDF-SHA256 (see [HPKE], Section 7.2)
- * AEAD: AES-128-GCM (see [HPKE], Section 7.3)

10. Experiment Overview

This document describes an experimental protocol built on DoH. The purpose of this experiment is to assess deployment configuration viability and related performance impacts on DNS resolution by measuring key performance indicators such as resolution latency. Experiment participants will test various parameters affecting service operation and performance, including mechanisms for discovery and configuration of DoH Proxies and Targets, as well as performance implications of connection reuse and pools where appropriate. The results of this experiment will be used to influence future protocol design and deployment efforts related to Oblivious DoH, such as Oblivious HTTP [OHTTP]. Implementations of DoH that are not involved in the experiment will not recognize this protocol and will not participate in the experiment. It is anticipated that use of Oblivious DoH and the duration of this experiment to be widespread.

11. Security Considerations

Oblivious DoH aims to keep knowledge of the true query origin and its contents known only to Clients. As a simplified model, consider a case where there exists two Clients C1 and C2, one proxy P, and one Target T. Oblivious DoH assumes an extended Dolev-Yao style attacker which can observe all network activity and can adaptively compromise either P or T, but not C1 or C2. Note that compromising both P and T is equivalent to collusion between these two parties in practice. Once compromised, the attacker has access to all session information and private key material. (This generalizes to arbitrarily many

Clients, Proxies, and Targets, with the constraints that not all Targets and Proxies are simultaneously compromised, and at least two Clients are left uncompromised.) The attacker is prohibited from sending Client identifying information, such as IP addresses, to Targets. (This would allow the attacker to trivially link a query to the corresponding Client.)

In this model, both C1 and C2 send an Oblivious DoH queries Q1 and Q2, respectively, through P to T, and T provides answers A1 and A2. The attacker aims to link C1 to (Q1, A1) and C2 to (Q2, A2), respectively. The attacker succeeds if this linkability is possible without any additional interaction. (For example, if T is compromised, it could return a DNS answer corresponding to an entity it controls, and then observe the subsequent connection from a Client, learning its identity in the process. Such attacks are out of scope for this model.)

Oblivious DoH security prevents such linkability. Informally, this means:

1. Queries and answers are known only to Clients and Targets in possession of the corresponding response key and HPKE keying material. In particular, Proxies know the origin and destination of an oblivious query, yet do not know the plaintext query. Likewise, Targets know only the oblivious query origin, i.e., the Proxy, and the plaintext query. Only the Client knows both the plaintext query contents and destination.
2. Target resolvers cannot link queries from the same Client in the absence of unique per-Client keys.

Traffic analysis mitigations are outside the scope of this document. In particular, this document does not prescribe padding lengths for ObliviousDoHQuery and ObliviousDoHResponse messages. Implementations SHOULD follow the guidance for choosing padding length in [RFC8467].

Oblivious DoH security does not depend on Proxy and Target indistinguishability. Specifically, an on-path attacker could determine whether a connection a specific endpoint is used for oblivious or direct DoH queries. However, this has no effect on confidentiality goals listed above.

11.1. Denial of Service

Malicious clients (or Proxies) can send bogus Oblivious DoH queries to targets as a Denial-of-Service (DoS) attack. Target servers can throttle processing requests if such an event occurs. Additionally, since Targets provide explicit errors upon decryption failure, i.e., if ciphertext decryption fails or if the plaintext DNS message is malformed, Proxies can throttle specific clients in response to these errors. In general, however, Targets trust Proxies to not overwhelm the Target, and it is expected that Proxies either implement some form of rate limiting or client authentication to limit abuse; see Section 11.3.

Malicious Targets or Proxies can send bogus answers in response to Oblivious DoH queries. Response decryption failure is a signal that either the Proxy or Target is misbehaving. Clients can choose to stop using one or both of these servers in the event of such failure. However, as above, malicious Targets and Proxies are out of scope for the threat model.

11.2. Proxy Policies

Proxies are free to enforce any forwarding policy they desire for Clients. For example, they can choose to only forward requests to known or otherwise trusted Targets.

Proxies that do not reuse connections to Targets for many Clients may allow Targets to link individual queries to unknown Targets. To mitigate this linkability vector, it is RECOMMENDED that Proxies pool and reuse connections to Targets. Note that this benefits performance as well as privacy since queries do not incur any delay that might otherwise result from Proxy-to-Target connection establishment.

11.3. Authentication

Depending on the deployment scenario, Proxies and Targets might require authentication before use. Regardless of the authentication mechanism in place, Proxies MUST NOT reveal any Client authentication information to Targets. This is required so Targets cannot uniquely identify individual Clients.

Note that if Targets require Proxies to authenticate at the HTTP- or application-layer before use, this ought to be done before attempting to forward any Client query to the Target. This will allow Proxies to distinguish 401 Unauthorized response codes due to authentication failure from 401 Unauthorized response codes due to Client key mismatch; see Section 4.3.

12. IANA Considerations

This document makes changes to the "Multipurpose Internet Mail Extensions (MIME) and Media Types" registry. The changes are described in the following subsection.

12.1. Oblivious DoH Message Media Type

This document registers a new media type, "application/oblivious-dns-message".

Type name: application

Subtype name: oblivious-dns-message

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: This is a binary format, containing encrypted DNS requests and responses encoded as ObliviousDoHMessage values, as defined in Section 6.1.

Security considerations: See this document. The content is an encrypted DNS message, and not executable code.

Interoperability considerations: This document specifies format of conforming messages and the interpretation thereof; see Section 6.1.

Published specification: This document.

Applications that use this media type: This media type is intended to be used by Clients wishing to hide their DNS queries when using DNS over HTTPS.

Additional information: N/A

Person and email address to contact for further information: See Authors' Addresses section

Intended usage: COMMON

Restrictions on usage: N/A

Author: Tommy Pauly tpauly@apple.com (mailto:tpauly@apple.com)

Change controller: IETF

Provisional registration? (standards tree only): No

13. Acknowledgments

This work is inspired by Oblivious DNS [I-D.annee-dprive-oblivious-dns]. Thanks to all of the authors of that document. Thanks to Nafeez Ahamed, Elliot Briggs, Marwan Fayed, Frederic Jacobs, Tommy Jensen, Jonathan Hoyland, Paul Schmitt, Brian Swander, Erik Nygren, and Peter Wu for the feedback and input.

14. References

14.1. Normative References

- [HPKE] Barnes, R. L., Bhargavan, K., Lipp, B., and C. A. Wood, "Hybrid Public Key Encryption", Work in Progress, Internet-Draft, draft-irtf-cfrg-hpke-12, 2 September 2021, <<https://www.ietf.org/archive/id/draft-irtf-cfrg-hpke-12.txt>>.
- [I-D.ietf-httpbis-proxy-status] Nottingham, M. and P. Sikora, "The Proxy-Status HTTP Response Header Field", Work in Progress, Internet-Draft, draft-ietf-httpbis-proxy-status-08, 13 October 2021, <<https://www.ietf.org/archive/id/draft-ietf-httpbis-proxy-status-08.txt>>.
- [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>>.
- [RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <<https://www.rfc-editor.org/info/rfc4086>>.
- [RFC6570] Gregorio, J., Fielding, R., Hadley, M., Nottingham, M., and D. Orchard, "URI Template", RFC 6570, DOI 10.17487/RFC6570, March 2012, <<https://www.rfc-editor.org/info/rfc6570>>.
- [RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <<https://www.rfc-editor.org/info/rfc7231>>.

- [RFC7235] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Authentication", RFC 7235, DOI 10.17487/RFC7235, June 2014, <<https://www.rfc-editor.org/info/rfc7235>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <<https://www.rfc-editor.org/info/rfc8446>>.
- [RFC8467] Mayrhofer, A., "Padding Policies for Extension Mechanisms for DNS (EDNS(0))", RFC 8467, DOI 10.17487/RFC8467, October 2018, <<https://www.rfc-editor.org/info/rfc8467>>.
- [RFC8484] Hoffman, P. and P. McManus, "DNS Queries over HTTPS (DoH)", RFC 8484, DOI 10.17487/RFC8484, October 2018, <<https://www.rfc-editor.org/info/rfc8484>>.

14.2. Informative References

- [I-D.annee-dprive-oblivious-dns] Edmundson, A., Schmitt, P., Feamster, N., and A. Mankin, "Oblivious DNS - Strong Privacy for DNS Queries", Work in Progress, Internet-Draft, draft-annee-dprive-oblivious-dns-00, 2 July 2018, <<https://www.ietf.org/archive/id/draft-annee-dprive-oblivious-dns-00.txt>>.
- [OHTP] Thomson, M. and C. A. Wood, "Oblivious HTTP", Work in Progress, Internet-Draft, draft-ietf-ohai-ohttp-01, 15 February 2022, <<https://www.ietf.org/archive/id/draft-ietf-ohai-ohttp-01.txt>>.
- [RFC7239] Petersson, A. and M. Nilsson, "Forwarded HTTP Extension", RFC 7239, DOI 10.17487/RFC7239, June 2014, <<https://www.rfc-editor.org/info/rfc7239>>.
- [RFC7871] Contavalli, C., van der Gaast, W., Lawrence, D., and W. Kumari, "Client Subnet in DNS Queries", RFC 7871, DOI 10.17487/RFC7871, May 2016, <<https://www.rfc-editor.org/info/rfc7871>>.

Appendix A. Use of Generic Proxy Services

Using DoH over anonymizing proxy services such as Tor can also achieve the desired goal of separating query origins from their contents. However, there are several reasons why such systems are undesirable in comparison Oblivious DoH:

1. Tor is meant to be a generic connection-level anonymity system, and incurs higher latency costs and protocol complexity for the purpose of proxying individual DNS queries. In contrast, Oblivious DoH is a lightweight protocol built on DoH, implemented as an application-layer proxy, that can be enabled as a default mode for users which need increased privacy.
2. As a one-hop proxy, Oblivious DoH encourages connection-less proxies to mitigate Client query correlation with few round-trips. In contrast, multi-hop systems such as Tor often run secure connections (TLS) end-to-end, which means that DoH servers could track queries over the same connection. Using a fresh DoH connection per query would incur a non-negligible penalty in connection setup time.

Authors' Addresses

Eric Kinnear
Apple Inc.
One Apple Park Way
Cupertino, California 95014,
United States of America
Email: ekinnear@apple.com

Patrick McManus
Fastly
Email: mcmanus@ducksong.com

Tommy Pauly
Apple Inc.
One Apple Park Way
Cupertino, California 95014,
United States of America
Email: tpauly@apple.com

Tanya Verma
Cloudflare
101 Townsend St
San Francisco,
United States of America
Email: vermatanyax@gmail.com

Christopher A. Wood
Cloudflare
101 Townsend St
San Francisco,
United States of America
Email: caw@heapingbits.net