

LISP Working Group  
Internet-Draft  
Intended status: Experimental  
Expires: November 30, 2019

S. Barkai  
B. Fernandez-Ruiz  
S. ZionB  
Nexar Inc

A. Rodriguez-Natal  
F. Maino  
Cisco Systems  
A. Cabellos-Aparicio  
J. Paillissé Vilanova  
Technical University of Catalonia  
D. Farinacci  
lispers.net  
June 30 2019

Network-Hexagons: H3-LISP Based Mobility Network  
draft-barkai-lisp-nexagon-05

Abstract

This document specifies combined use of H3 and LISP for mobility-networks:  
- Enabling real-time localized indexed-annotation of roads, tile by tile  
- For sharing - hazards, blockages, conditions, maintenance, furniture..  
- Between MobilityClients producing and consuming road-state information  
- Via in-network-state, an indirection by an addressable physical world grid.

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 October 4, 2019.

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
2. Requirements Language . . . . .	3
3. Definition of Terms . . . . .	3
4. Deployment Assumptions . . . . .	4

5. Mobility Clients-Network-Services . . . . .	4
6. Mobility Unicast-Multicast . . . . .	5
7. Security Considerations . . . . .	6
8. Acknowledgments . . . . .	6
9. IANA Considerations . . . . .	6
10. Normative References . . . . .	8
Authors' Addresses . . . . .	9

## 1. Introduction

(1) The Locator/ID Separation Protocol (LISP) [RFC6830] splits current IP addresses in two different namespaces, Endpoint Identifiers (EIDs) and Routing Locators (RLOCs). LISP uses a map-and-encap approach that relies on (1) a Mapping System (distributed database) that stores and disseminates EID-RLOC mappings and on (2) LISP tunnel routers (xTRs) that encapsulate and decapsulate data packets based on the content of those mappings.

(2) H3 is a geospatial indexing system using a hexagonal grid that can be (approximately) subdivided into finer and finer hexagonal grids, combining the benefits of a hexagonal grid with hierarchical subdivisions. H3 supports sixteen resolutions. Each finer resolution has cells with one seventh the area of the coarser resolution. Hexagons cannot be perfectly subdivided into seven hexagons, so the finer cells are only approximately contained within a parent cell. Each cell is identified by a 64bit HID.

(3) The Berkeley Deep Drive (BDD) Industry Consortium investigates state-of-the-art technologies in computer vision and machine learning for automotive applications, and, for taxonomy of published automotive scene classification.

These standards are combined to create in-network-state which reflects the condition of each one-square-meter (~1sqm) hexagon road-tile. The lisp network maps & encapsulates traffic between MobilityClients endpoint-identifiers (EID, and, addressable (HID=>EID) tile-states, aggregated by H3Service EIDs.

The H3-LISP mobility network bridges timing-location gaps between the production and consumption of information by MobilityClients:  
 - vision, sensory, LIADR, AI applications - info-producers  
 - driving-apps, smart-infrastructure, command & control - info-consumers  
 This is achieved by putting the physical world on a shared addressable state-grid, an indirection.

The tile based geo-state mobility-network solves key issues in todays' vehicle to vehicle networking, where observed hazards are expected to be relayed or "hot-potato-tossed" (v2v) between vehicles without clear convergence i.e. given a situation observable by some of traffic it is unclear if the rest of the relevant traffic will receive consistent, conflicting, multiple, or non what so ever peer-to-peer v2v indication.

For example, when a vehicle experiences a sudden highway slow-down, "sees" many brake-lights or "feels" accelerometer, there is no clear way for it to share this annotation with vehicles 20-30 sec away, preventing potential pile-up. Or, when a vehicle crosses an intersection, observing opposite-lane obstruction - construction, double-park, commercial-loading / un-loading, garbage truck, or stopped school-bus - there is no clear way for it to alert vehicles turning in to that lane - as it crossed and drove away. Data may be replicated distorted or lost just like in a telephone-game.

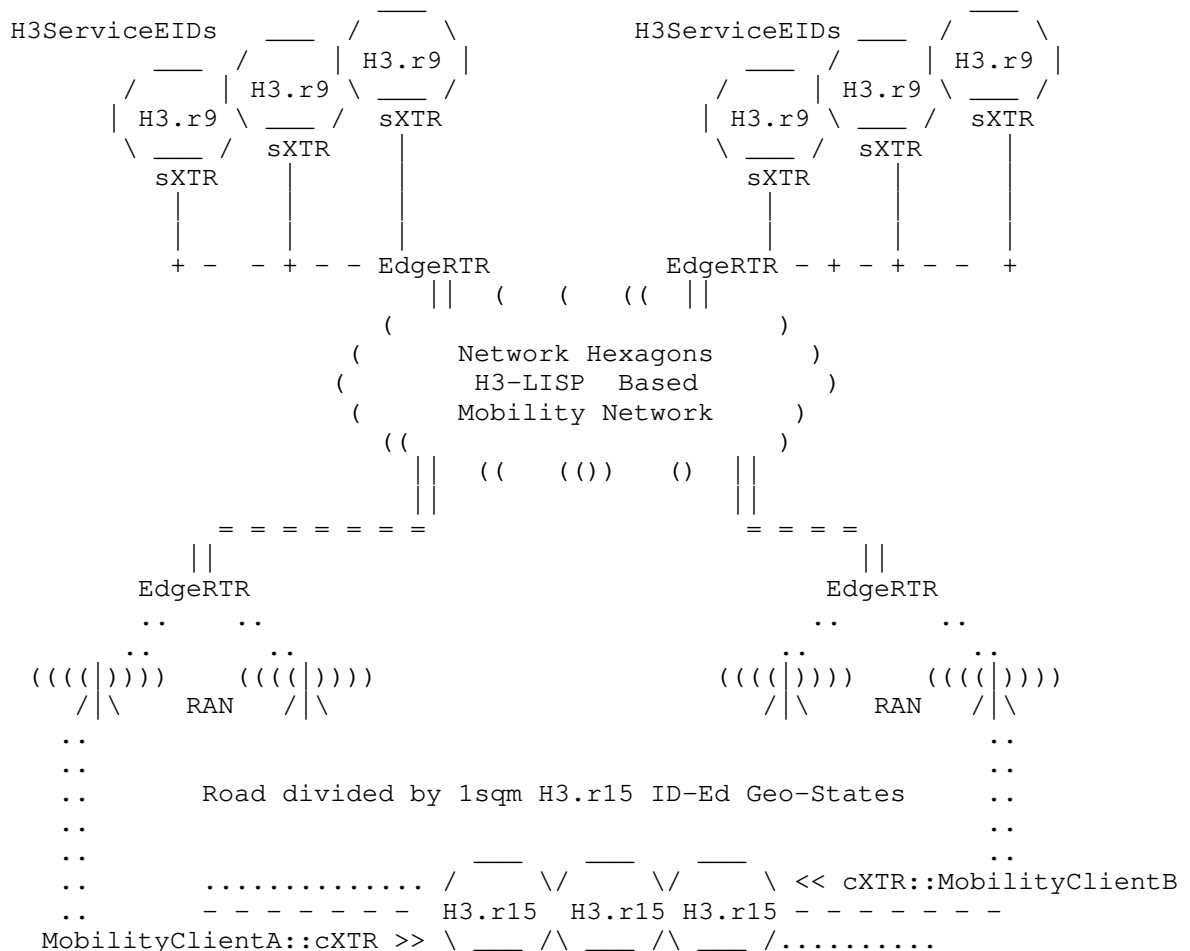
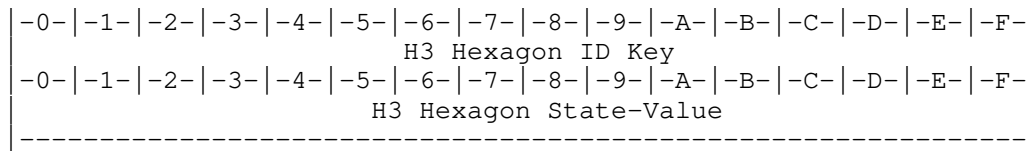
These limitations are inherit since in most road situations vehicles are not really proper peers. They just happen to be in the same place at the same time

The H3-LISP mobility network solves limitations of direct vehicle to vehicle communication because it anchors per this place: timing, privacy, interoperability. This anchoring is done by MobilityClients (EIDs) communicating through in-network road-tile geo-states. Geo-states are aggregated and maintained by LISP addressable H3ServiceEIDs.

An important set of use-cases for state propagation of information to MobilityClients is to provide drivers heads-up alerts on hazards and obstacles beyond line of sight of both the drivers and in-car sensors: over traffic, around blocks, far-side-junction, beyond turns, and surface-curvatures. This highlights the importance of networks in providing road-safety.

To summarize the H3-LISP solution outline:

- (1) Partition: 64bit indexed geo-spatial H3.r15 (~1sqm) road-tiles
- (2) State: 64bit state values compile tile condition representation
- (3) Aggregation: H3.r9 H3ServiceEID group individual H3.r15 road-tiles
- (4) Channels: H3ServiceEIDs function as multicast state update channels
- (5) Scale: H3ServiceEIDs distributed for in-network for latency-throughput
- (6) Mapped Overlay: tunneled-network routes the mobility-network traffic
- (7) Signal-free: tunneled overlay is used to map-register for mcast channels
- (8) Access: tunnels used between MobilityClients/H3ServiceEIDs <> LISP edge
- (9) Access: ClientXTRs/ServerXTRs tunnel traffic to-from the LISP EdgeRTRs
- (10) Control: EdgeRTRs register-resolve identity-location and mcast (s,g)



- MobilityClientA has seen MobilityClientB (20-30 sec) future, and, vice versa
- Clients share information using addressable shared-state routed by LISP Edge
- ClientXTR (cXTR): tunnel encapsulation through access network to LISP Edge

- ServerXTR (sXTR): tunnel encapsulation through cloud network to LISP Edge
- The H3-LISP Mobility overlay starts in the cXTR and terminates in the sXTR
- The updates are routed to the appropriate tile geo-state by the LISP network
- EdgeRTRs perform multicast replication to edges and then native or to cXTRs
- Clients receive tile-by-tile geo-state updates via the multicast channels

Each H3.r9 hexagon is an EID Service with corresponding H3 hexagon ID. Bound to that service is a LISP xTR, called a ServerXTR, resident to deliver encapsulated packets to and from the H3ServiceEID and LISP Edge. EdgeRTRs are used to re-tunnel packets from MobilityClients to H3ServiceEIDs. Each H3ServiceEID is also a source multicast address for updating MobilityClients on the state of the H3.r15 tiles aggregated-represented by the H3ServiceEID.

## 2. 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 [RFC2119].

## 3. Definition of Terms

**H3ServiceEID:** Is an addressable aggregation of H3.r15 state-tiles. It is a designated source for physical world reported annotations, and an (s,g) source of multicast public-safety update channels. H3ServiceEID is itself an H3 hexagon, large enough to provide geo-spatial conditions context, but not too large as to over-burden (battery powered, cellular connected) subscribers with too much information. For Mobility Network it is H3.r9. It has a light-weight LISP protocol stack to tunnel packets aka ServerXTR. The EID is an IPv6 EID that contains the H3 64-bit address numbering scheme. See IANA consideration for details.

**ServerXTR:** Is a light-weight LISP protocol stack implementation that co-exists with H3ServiceEID process. When the server roams, the xTR roams with it. The ServerXTR encapsulates and decapsulates packets to/from EdgeRTRs.

**MobilityClient:** Is a roaming application that may be resident as part of an automobile, as part of a navigation application, part of municipal, state, or federal government command and control application, or part of live street view consumer type of application. It has a light-weight LISP protocol stack to tunnel packets aka ClientXTR.

**MobilityClient EID:** Is the IPv6 EID used by the Mobility Client applications to source packets. The destination of such packets are only H3ServiceEIDs. The EID format is opaque and is assigned as part of the MobilityClient network-as-a-service (Naas) authorization.

**ClientXTR:** Is the light-weight LISP protocol stack implementation that is co-located with the Mobility Client application. It encapsulates packets sourced by applications to EdgeRTRs and decapsulates packets from EdgeRTRs.

**EdgeRTR:** Is the core scale and structure of the LISP mobility network. LISP RTRs decapsulate packets from ClientXTRs and ServerXTRs and re-encapsulates packets to ServerXTRs and ClientXTRs. The EdgeRTRs glean H3ServiceEIDs and glean MobilityClient EIDs when it decapsulates packets. EdgeRTRs store H3ServiceEIDs and their own RLOC of where the H3ServiceEID is currently reachable from in the map-cache. These mappings are registered to the LISP mapping system so other EdgeRTRs know where to encapsulate for such EIDs.

## 4. Deployment Assumptions

The specification described in this document makes the following deployment assumptions:

- (1) Unique 64-bit HID is associated with each H3 geo-spatial tile
- (2) MobilityClients and H3ServiceEIDs share this well known index
- (3) 64-bit BDD state value is associated with each H3-indexed tile
- (4) Tile state is compiled 16 fields of 4-bits, or max 16 enums

```
| -0- | -1- | -2- | -3- | -4- | -5- | -6- | -7- | -8- | -9- | -A- | -B- | -C- | -D- | -E- | -F- |
01230123012301230123012301230123012301230123012301230123012301230123012301230123
```

A MobilityClient which needs to use an H3-LISP mobility overlay network - instantiates a ClientXTR. It leverages DNS resolution to find EdgeRTR(s) in order to home to. ClientXTR is provisioned with an address for DNS resolvers, that help with the EdgeRTR discovery. The ClientXTR uses these DNS resolvers to resolve a query that includes the ClientXTRs current H3 index at resolution 9 (e.g. h3res9.example.net). To find its current H3.r9 index, the ClientXTR first translates its current geo-location to an H3 index (e.g. gps snap-to-h3.r9). As a response to the query including the H3.r9 index of the ClientXTR, the DNS resolver will return the IP address of the Edge RTR that the ClientXTR can use to home to the H3-LISP mobility overlay.

The EdgeRTR discovery by the ClientXTR performed via DNS resolution so that:

- 1) EdgeRTRs are not tightly coupled to H3.r9 areas for easy load-balance
- 2) Mobility Clients do not need to constantly update EdgeRTR when it roams

In that sense, the same EdgeRTR may serve several H3.r9 areas for smooth ride continuity, and, several EdgeRTRs may load balance a H3.r9 area with high density of originating MobilityClient rides. When a MobilityClient ClientXTR is homed to EdgeRTR it is able to communicate with H3ServiceEIDs.

## 5. Mobility Clients-Network-Services

The mobility network functions as a standard LISP VPN overlay. The overlay delivers unicast and multicast packets across:

- multiple access-network-providers / radio-access-technologies.
- multiple cloud-edge hosting providers, public, private, hybrid.

We use data-plane XTRs in the stack of each mobility client and server. ClientXTRs and ServerXTRs are homed to one or more EdgeRTRs at the LISP edge. This structure allows for MobilityClients to "show-up" at any time, behind any network-provider in a given mobility network administrative domain (metro), and for any H3ServiceEID to be instantiated, moved, or failed-over to - any rack in any cloud-provider. The LISP overlay enables these roaming mobility network elements to communicate un-interrupted. This quality is insured by the LISP RFCs. The determinism of identities for MobilityClients to always refer to the correct H3ServiceEID is insured by H3 geospatial HIDs.

There are two options for how we associate ClientXTRs with LISP EdgeRTRs:

### I. semi-random through DNS based load-balancing

In this option we assume that in a given metro edge a pool of EdgeRTRs can distribute the Mobility Clients load randomly between them and that EdgeRTRs are topologically more or less equivalent. Each RTR uses LISP to mesh with the other RTRs in order to connect each Mobility Client with H3 Servers. Mobility Clients can (multi) home to the same RTR(s) throughout a ride.

### II. geo-spatial, where a well known any-cast RTR aggregates H3.r9 hexagons

In this option we align an EdgeRTR with a geo-spatial cell area, very much like in Evolved Packet Core (EPC) solution. Mobility Clients currently roaming in an area home to that RTR and so is the H3 Server. There is only one hop across the edge overlay between clients and servers and mcast replication is more focused, but clients need to keep re-homing as they move.

To summarize the H3LISP mobility network layout:

- (1) Mobility-Clients traffic is tunneled via data-plane ClientXTRs  
ClientXTRs are (multi) homed to EdgeRTR(s)
- (2) H3ServiceEID traffic is tunneled via data-plane ServerXTR  
ServerXTRs are (multi) homed to EdgeRTR(s)
- (3) EdgeRTRs use mapping service to resolve Ucast HIDs to RTR RLOCs  
EdgeRTRs also register to (Source, Group) H3ServiceEID multicasts

```

MobilityClients <> ClientXTR <Access Provider > EdgeRTR  v
                                                         v
v      << Map-Assisted Mobility-Network Overlay <<      v
v
>> EdgeRTR <Cloud Provider> ServerXTR <> H3ServiceEID

```

## 6. Mobility Unicast and Multicast

Which ever way a ClientXTR is homed to an Edge RTR, DNS metro load-balance or well known geo-spatial map of IPs (a few 10Ks per large metro area), an authenticated MobilityClient EID can send: [64bitH3.15ID :: 64bitState] annotation to the H3.r9 H3ServiceEID. The H3.r9 IP HID can be calculated by clients algorithmically form the H3.15 localized snapped-to-tile annotation.

The ClientXTR encapsulates MobilityClient EID and H3ServiceEID in a packet sourced from the ClientXTR, destined to the EdgeRTR RLOC IP, Lisp port. EdgeRTRs then re-encapsulate annotation packets either to remote EdgeRTR (option1) or to homed H3ServiceEID ServerXTR (option2). The remote EdgeRTR aggregating H3ServiceEIDs re-encapsulates MobilityClient EID to ServerXTR and from there to the H3ServiceEID.

To Summarize Unicast:

- (1) MobilityClients can send annotation state localized an H3.r15 tile  
These annotations are sent to an H3.r9 mobility H3ServiceEIDs
- (2) MobilityClient EID and H3ServiceEID HID are encapsulated:  
XTR <> RTR <> RTR <> XTR  
\* RTRs can map-resolve re-tunnel HIDs
- (3) RTRs re-encapsulate original source-dest to ServerXTRs  
ServerXTRs decapsulate packets to H3ServiceEID

Each H3.r9 Server is used by clients to update H3.r15 tile state is also an IP Multicast channel Source used to update subscribers on the aggregate state of the H3.r15 tiles in the H3.r9 Server.

We use rfc8378 signal free multicast to implement mcast channels in the overlay. The mobility network has many channels and relatively few subscribers per each. MobilityClients driving through or subscribing to a H3.r9 area can explicitly issue an rfc4604 MLDv2 in-order to subscribe, or, may be subscribed implicitly by the EdgeRTR gleaning to ucast HID dest.

The advantage of explicit client MLDv2 registration trigger to rfc8378 is that the clients manage their own mobility mcast hand-over according to their location-direction moment vectors, and that it allows for otherwise silent, or

, non annotating clients. The advantage of EdgeRTR implicit registration is less signaling required.

MLDv2 signaling messages are encapsulated between the ClientXTR and the LISP EdgeRTR, therefore there is no requirement for the underlying network to support native multicast. If native access multicast is supported (for example native 5G multicast), then MobilityClient registration to H3ServiceEID safety channels may be integrated to it, in which case the evolved-packet-core (EPC) element supporting it (eNB) will use this standard to register with the appropriate H3.r9 channels in its area.

EdgeRTRs note the subscribed MobilityClient stack XTRs and register as channel subscribers in the mapping system (Source, Group) entry. This is done at the first subscription request, if additional MobilityClients homed to the same EdgeRTR register for the same channels the EdgeRTR registration covers them.

Upon receiving a multicast packet the EdgeRTR homing H3.r9 Servers resolve the (S,G) remote EdgeRTRs registered for the channel and replicates the packet

- The remote EdgeRTRs homing MobilityClients in-turn replicate the packet to the MobilityClients registered with them.

We expect an average of 600 H3.r15 tiles of the full  $7^6$  (~100K) possible in H3.r9 to be part of any road. The H3.r9 server can transmit the status of all 600 or just those with meaningful state based on update SLA and policy.

To Summarize:

- (1) H3LISP Clients tune to H3.r9 mobility updates using rfc8378  
H3LISP Client issue MLDv2 registration to H3.r9 HIDs  
ClientXTRs encapsulate MLDv2 to EdgeRTRs who register (s,g)
- (2) ServerXTRs encapsulate updates to EdgeRTRs who map-resolve (s,g) RLOCs  
EdgeRTRs replicate mobility update and tunnel to registered EdgeRTRs  
Remote EdgeRTRs replicate updates to registered ClientXTRs

## 7. Security Considerations

The way to provide a security association between the ITRs and the Map-Servers must be evaluated according to the size of the deployment. For small deployments, it is possible to have a shared key (or set of keys) between the ITRs and the Map-Servers. For larger and Internet-scale deployments, scalability is a concern and further study is needed.

## 8. Acknowledgments

This work is partly funded by the ANR LISP-Lab project #ANR-13-INFR-009 (<https://lisplab.lip6.fr>).

## 9. IANA Considerations

### I. Formal H3 to IPv6 EID mapping

### II. State enum fields of H3 tiles:

Field 0x describes the "freshness" of the state {

0x: less than 1Sec  
1x: less than 10Sec  
2x: less than 20Sec  
3x: less than 40Sec  
4x: less than 1min  
5x: less than 2min  
6x: less than 5min  
7x: less than 15min  
8x: less than 30min  
9x: less than 1hour  
Ax: less than 2hours  
Bx: less than 8hours  
Cx: less than 24hours  
Dx: less than 1week  
Ex: less than 1month  
Fx: more than 1month

```

}

field 1x: persistent weather or structural {
  0x - null
  1x - pothole
  2x - speed-bump
  3x - icy
  4x - flooded
  5x - snow-cover
  6x - snow-deep
  7x - construction cone
  8x - curve
}

field 2x: transient or moving obstruction {
  0x - null
  1x - pedestrian
  2x - bike
  3x - stopped car / truck
  4x - moving car / truck
  5x - first responder vehicle
  6x - sudden slowdown
  7x - oversized-vehicle
  8x - red-light-breach
}

field 3x: traffic-light timer countdown {
  0x - green now
  1x - 1 seconds to green
  2x - 2 seconds to green
  3x - 3 seconds to green
  4x - 4 seconds to green
  5x - 5 seconds to green
  6x - 6 seconds to green
  7x - 7 seconds to green
  8x - 8 seconds to green
  9x - 9 seconds to green
  Ax - 10 seconds or less
  Bx - 20 seconds or less
  Cx - 30 seconds or less
  Dx - 40 seconds or less
  Ex - 50 seconds or less
  Fx - minute or more left
}

field 4x: impacted tile from neighboring {
  0x - not impacted
  1x - light yellow
  2x - yellow
  3x - light orange
  4x - orange
  5x - light red
  6x - red
  7x - light blue
  8x - blue
}

field 5x: incidents {
  0x - clear
  1x - light collision (fender bender)
  2x - hard collision
  3x - collision with casualty
  4x - recent collision residues
  5x - hard brake
  6x - sharp cornering
}

field 6x - compiled tile safety rating {

```



```

}
field 7x: LaneRightsSigns {
    0x - stop
    1x - yield
    2x - speedLimit
    3x - straightOnly
    4x - noStraight
    5x - rightOnly
    6x - noRight
    7x - leftOnly
    8x - noLeft
    9x - noUTurn
    10x - noLeftU
    11x - bikeLane
    12x - HOVLane
}

field 8x: MovementSigns {
    0x - noPass
    1x - keepRight
    2x - keepLeft
    3x - stayInLane
    4x - doNotEnter
    5x - noTrucks
    6x - noBikes
    7x - noPeds
    8x - oneWay
    9x - parking
    10x - noParking
    11x - noStandaing
    12x - loadingZone
    13x - truckRoute
    14x - railCross
    15x - School
}

field 9x: CurvesIntersectSigns {
    0x - turnsLeft
    1x - turnsRight
    2x - curvesLeft
    3x - curvesRight
    4x - reversesLeft
    5x - reversesRight
    6x - windingRoad
    7x - hairPin
    8x - 270Turn
    9x - pretzelTurn
    10x - crossRoads
    11x - crossT
    12x - crossY
    13x - circle
    14x - laneEnds
    15x - roadNarrows
}
field Ax - reserved
field Bx - reserved
field Cx - reserved
field Dx - reserved
field Ex - reserved
field Fx - reserved

```

[I-D.ietf-lisp-rfc6833bis]

Fuller, V., Farinacci, D., and A. Cabellos-Aparicio,  
"Locator/ID Separation Protocol (LISP) Control-Plane",  
draft-ietf-lisp-rfc6833bis-07 (work in progress), December  
2017.

[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>>.

[RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The  
Locator/ID Separation Protocol (LISP)", RFC 6830,  
DOI 10.17487/RFC6830, January 2013,  
<<https://www.rfc-editor.org/info/rfc6830>>.

[RFC8378] Farinacci, D., Moreno, V., "Signal-Free Locator/ID Separation  
Protocol (LISP) Multicast", RFC8378,  
DOI 10.17487/RFC8378, May 2018,  
<<https://www.rfc-editor.org/info/rfc8378>>.

#### Authors' Addresses

Sharon Barkai  
Nexar  
CA  
USA

Email: [sbarkai@gmail.com](mailto:sbarkai@gmail.com)

Bruno Fernandez-Ruiz  
Nexar  
London  
UK

Email: [b@getnexar.com](mailto:b@getnexar.com)

S ZionB  
Nexar  
Israel

Email: [sharon@fermicloud.io](mailto:sharon@fermicloud.io)

Alberto Rodriguez-Natal  
Cisco Systems  
170 Tasman Drive  
San Jose, CA  
USA

Email: [natal@cisco.com](mailto:natal@cisco.com)

Fabio Maino  
Cisco Systems  
170 Tasman Drive  
San Jose, CA  
USA

Email: [fmaino@cisco.com](mailto:fmaino@cisco.com)

Albert Cabellos-Aparicio  
Technical University of Catalonia  
Barcelona  
Spain

Email: [acabello@ac.upc.edu](mailto:acabello@ac.upc.edu)

Jordi Paillissé-Vilanova  
Technical University of Catalonia  
Barcelona  
Spain

Email: [jordip@ac.upc.edu](mailto:jordip@ac.upc.edu)

Dino Farinacci  
[lispsers.net](http://lispsers.net)  
San Jose, CA  
USA

Email: [farinacci@gmail.com](mailto:farinacci@gmail.com)

TSVWG  
Internet-Draft  
Intended status: Informational  
Expires: January 29, 2021

A. Ferrieux, Ed.  
I. Hamchaoui, Ed.  
Orange Labs  
I. Lubashev, Ed.  
Akamai Technologies  
D. Tikhonov, Ed.  
LiteSpeed Technologies  
July 28, 2020

Packet Loss Signaling for Encrypted Protocols  
draft-ferrieuxhamchaoui-tsvwg-lossbits-03

Abstract

This document describes a protocol-independent method that employs two bits to allow endpoints to signal packet loss in a way that can be used by network devices to measure and locate the source of the loss. The signaling method applies to all protocols with a protocol-specific way to identify packet loss. The method is especially valuable when applied to protocols that encrypt transport header and do not allow an alternative method for loss detection.

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 January 29, 2021.

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 . . . . .	3
1.1. Motivation for Passive On-Path Loss Observation . . . . .	3
1.2. On-Path Loss Observation . . . . .	3
1.3. On-Path Loss Signaling . . . . .	4
1.4. Recommended Use of the Signals . . . . .	4
2. Notational Conventions . . . . .	4
3. Loss Bits . . . . .	4
3.1. Setting the sSquare Bit on Outgoing Packets . . . . .	5
3.1.1. Q Run Length Selection . . . . .	5
3.2. Setting the Loss Event Bit on Outgoing Packets . . . . .	5
4. Using the Loss Bits for Passive Loss Measurement . . . . .	6
4.1. End-To-End Loss . . . . .	6
4.2. Upstream Loss . . . . .	6
4.3. Correlating End-to-End and Upstream Loss . . . . .	7
4.4. Downstream Loss . . . . .	7
4.5. Observer Loss . . . . .	7
5. ECN-Echo Event Bit . . . . .	8
5.1. Setting the ECN-Echo Event Bit on Outgoing Packets . . . . .	8
5.2. Using E Bit for Passive ECN-Reported Congestion Measurement . . . . .	9
6. Protocol Ossification Considerations . . . . .	9
7. Security Considerations . . . . .	9
7.1. Optimistic ACK Attack . . . . .	10
8. Privacy Considerations . . . . .	10
9. IANA Considerations . . . . .	10
10. Change Log . . . . .	10
10.1. Since version 02 . . . . .	10
10.2. Since version 01 . . . . .	11
10.3. Since version 00 . . . . .	11
11. Acknowledgments . . . . .	11
12. References . . . . .	11
12.1. Normative References . . . . .	11
12.2. Informative References . . . . .	12
Authors' Addresses . . . . .	13

## 1. Introduction

### 1.1. Motivation for Passive On-Path Loss Observation

Packet loss is hard and pervasive problem of day-to-day network operation. Proactively detecting, measuring, and locating it is crucial to maintaining high QoS and timely resolution of crippling end-to-end throughput issues. To this effect, in a TCP-dominated world, network operators have been heavily relying on information present in the clear in TCP headers: sequence and acknowledgment numbers and SACKs when enabled (see [RFC8517]). These allow for quantitative estimation of packet loss by passive on-path observation. Additionally, the lossy segment (upstream or downstream from the observation point) can be quickly identified by moving the passive observer around.

With encrypted protocols, the equivalent transport headers are encrypted and passive packet loss observation is not possible, as described in [TRANSPORT-ENCRYPT].

Measuring TCP loss between similar endpoints cannot be relied upon to evaluate encrypted protocol loss. Different protocols could be routed by the network differently and the fraction of Internet traffic delivered using protocols other than TCP is increasing every year. It is imperative to measure packet loss experienced by encrypted protocol users directly.

### 1.2. On-Path Loss Observation

There are three sources of loss that network operators need to observe to guarantee high QoS:

- `_upstream loss_` - loss between the sender and the observation point (Section 4.2)
- `_downstream loss_` - loss between the observation point and the destination (Section 4.4)
- `_observer loss_` - loss by the observer itself that does not cause downstream loss (Section 4.5)

The upstream and downstream loss together constitute `_end-to-end loss_` (Section 4.1).

### 1.3. On-Path Loss Signaling

Following the recommendation in [RFC8558] of making path signals explicit, this document proposes adding two explicit loss bits to the clear portion of the protocol headers to restore network operators' ability to maintain high QoS. These bits can be added to an unencrypted portion of a header belonging to any protocol layer, e.g. IP (see [IP]) and IPv6 (see [IPv6]) headers or extensions, such as [IPv6AltMark], UDP surplus space (see [UDP-OPTIONS] and [UDP-SURPLUS]), reserved bits in a QUIC v1 header (see [QUIC-TRANSPORT]).

### 1.4. Recommended Use of the Signals

The loss signal is not designed for use in automated control of the network in environments where loss bits are set by untrusted hosts. Instead, the signal is to be used for troubleshooting individual flows as well as for monitoring the network by aggregating information from multiple flows and raising operator alarms if aggregate statistics indicate a potential problem.

## 2. Notational Conventions

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

## 3. Loss Bits

The draft introduces two bits that are to be present in packets capable of loss reporting. These are packets that include protocol headers with the loss bits. Only loss of packets capable of loss reporting is reported using loss bits.

Whenever this specification refers to packets, it is referring only to packets capable of loss reporting.

- Q: The "sQuare signal" bit is toggled every N outgoing packets as explained below in Section 3.1.
- L: The "Loss event" bit is set to 0 or 1 according to the Unreported Loss counter, as explained below in Section 3.2.

Each endpoint maintains appropriate counters independently and separately for each separately identifiable flow (each subflow for multipath connections).

### 3.1. Setting the sSquare Bit on Outgoing Packets

The sSquare Value is initialized to the Initial Q Value (0) and is reflected in the Q bit of every outgoing packet. The sSquare value is inverted after sending every N packets (a Q Run). Hence, Q Period is  $2*N$ . The Q bit represents "packet color" as defined by [RFC8321]. The sSquare Bit can also be called an Alternate Marking bit.

Observation points can estimate the upstream losses by counting the number of packets during a half period of the square signal, as described in Section 4.

#### 3.1.1. Q Run Length Selection

The sender is expected to choose N (Q run length) based on the expected amount of loss and reordering on the path. The choice of N strikes a compromise - the observation could become too unreliable in case of packet reordering and/or severe loss if N is too small, while short flows may not yield a useful upstream loss measurement if N is too large (see Section 4.2).

The value of N MUST be at least 64 and be a power of 2. This requirement allows an Observer to infer the Q run length by observing one period of the square signal. It also allows the Observer to identify flows that set the loss bits to arbitrary values (see Section 6).

If the sender does not have sufficient information to make an informed decision about Q run length, the sender SHOULD use  $N=64$ , since this value has been extensively tried in large-scale field tests and yielded good results. Alternatively, the sender MAY also choose a random N for each flow, increasing the chances of using a Q run length that gives the best signal for some flows.

The sender MUST keep the value of N constant for a given flow.

### 3.2. Setting the Loss Event Bit on Outgoing Packets

The Unreported Loss counter is initialized to 0, and L bit of every outgoing packet indicates whether the Unreported Loss counter is positive ( $L=1$  if the counter is positive, and  $L=0$  otherwise). The value of the Unreported Loss counter is decremented every time a packet with  $L=1$  is sent.

The value of the Unreported Loss counter is incremented for every packet that the protocol declares lost, using whatever loss detection machinery the protocol employs. If the protocol is able to rescind the loss determination later, a positive Unreported Loss counter MAY



be decremented due to the rescission, but it SHOULD NOT become negative due to the rescission.

This loss signaling is similar to loss signaling in [ConEx], except the Loss Event bit is reporting the exact number of lost packets, whereas Echo Loss bit in [ConEx] is reporting an approximate number of lost bytes.

For protocols, such as TCP ([TCP]), that allow network devices to change data segmentation, it is possible that only a part of the packet is lost. In these cases, the sender MUST increment Unreported Loss counter by the fraction of the packet data lost (so Unreported Loss counter may become negative when a packet with L=1 is sent after a partial packet has been lost).

Observation points can estimate the end-to-end loss, as determined by the upstream endpoint, by counting packets in this direction with the L bit equal to 1, as described in Section 4.

#### 4. Using the Loss Bits for Passive Loss Measurement

##### 4.1. End-To-End Loss

The Loss Event bit allows an observer to calculate the end-to-end loss rate by counting packets with L bit value of 0 and 1 for a given flow. The end-to-end loss rate is the fraction of packets with L=1.

The assumption here is that upstream loss affects packets with L=0 and L=1 equally. If some loss is caused by tail-drop in a network device, this may be a simplification. If the sender's congestion controller reduces the packet send rate after loss, there may be a sufficient delay before sending packets with L=1 that they have a greater chance of arriving at the observer.

##### 4.2. Upstream Loss

Blocks of N (Q Run length) consecutive packets are sent with the same value of the Q bit, followed by another block of N packets with an inverted value of the Q bit. Hence, knowing the value of N, an on-path observer can estimate the amount of upstream loss after observing at least N packets. The upstream loss rate ("u") is one minus the average number of packets in a block of packets with the same Q value ("p") divided by N ( $u = 1 - \text{avg}(p)/N$ ).

The observer needs to be able to tolerate packet reordering that can blur the edges of the square signal.

The observer needs to differentiate packets as belonging to different flows, since they use independent counters.

#### 4.3. Correlating End-to-End and Upstream Loss

Upstream loss is calculated by observing packets that did not suffer the upstream loss. End-to-end loss, however, is calculated by observing subsequent packets after the sender's protocol detected the loss. Hence, end-to-end loss is generally observed with a delay of between 1 RTT (loss declared due to multiple duplicate acknowledgments) and 1 RTO (loss declared due to a timeout) relative to the upstream loss.

The flow RTT can sometimes be estimated by timing protocol handshake messages. This RTT estimate can be greatly improved by observing a dedicated protocol mechanism for conveying RTT information, such as the Latency Spin bit of [QUIC-TRANSPORT].

Whenever the observer needs to perform a computation that uses both upstream and end-to-end loss rate measurements, it SHOULD use upstream loss rate leading the end-to-end loss rate by approximately 1 RTT. If the observer is unable to estimate RTT of the flow, it should accumulate loss measurements over time periods of at least 4 times the typical RTT for the observed flows.

If the calculated upstream loss rate exceeds the end-to-end loss rate calculated in Section 4.1, then either the Q Period is too short for the amount of packet reordering or there is observer loss, described in Section 4.5. If this happens, the observer SHOULD adjust the calculated upstream loss rate to match end-to-end loss rate.

#### 4.4. Downstream Loss

Because downstream loss affects only those packets that did not suffer upstream loss, the end-to-end loss rate ("e") relates to the upstream loss rate ("u") and downstream loss rate ("d") as  $(1-u)(1-d)=1-e$ . Hence,  $d=(e-u)/(1-u)$ .

#### 4.5. Observer Loss

A typical deployment of a passive observation system includes a network tap device that mirrors network packets of interest to a device that performs analysis and measurement on the mirrored packets. The observer loss is the loss that occurs on the mirror path.

Observer loss affects upstream loss rate measurement since it causes the observer to account for fewer packets in a block of identical Q

bit values (see {{upstreamloss}}). The end-to-end loss rate measurement, however, is unaffected by the observer loss, since it is a measurement of the fraction of packets with the set L bit value, and the observer loss would affect all packets equally (see Section 4.1).

The need to adjust the upstream loss rate down to match end-to-end loss rate as described in Section 4.3 is a strong indication of the observer loss, whose magnitude is between the amount of such adjustment and the entirety of the upstream loss measured in Section 4.2. Alternatively, a high apparent upstream loss rate could be an indication of significant reordering, possibly due to packets belonging to a single flow being multiplexed over several upstream paths with different latency characteristics.

## 5. ECN-Echo Event Bit

While the primary focus of the draft is on exposing packet loss, modern networks can report congestion before they are forced to drop packets, as described in [ECN]. When transport protocols keep ECN-Echo feedback under encryption, this signal cannot be observed by the network operators. When tasked with diagnosing network performance problems, knowledge of a congestion downstream of an observation point can be instrumental.

If downstream congestion information is desired, this information can be signaled with an additional bit.

- E: The "ECN-Echo Event" bit is set to 0 or 1 according to the Unreported ECN Echo counter, as explained below in Section 5.1.

### 5.1. Setting the ECN-Echo Event Bit on Outgoing Packets

The Unreported ECN-Echo counter operates identically to Unreported Loss counter (Section 3.2), except it counts packets delivered by the network with CE markings, according to the ECN-Echo feedback from the receiver.

This ECN-Echo signaling is similar to ECN signaling in [ConEx]. ECN-Echo mechanism in QUIC provides the number of packets received with CE marks. For protocols like TCP, the method described in [ConEx-TCP] can be employed. As stated in [ConEx-TCP], such feedback can be further improved using a method described in [ACCURATE].

## 5.2. Using E Bit for Passive ECN-Reported Congestion Measurement

A network observer can count packets with CE codepoint and determine the upstream CE-marking rate directly.

Observation points can also estimate ECN-reported end-to-end congestion by counting packets in this direction with a E bit equal to 1.

The upstream CE-marking rate and end-to-end ECN-reported congestion can provide information about downstream CE-marking rate. Presence of E bits along with L bits, however, can somewhat confound precise estimates of upstream and downstream CE-markings in case the flow contains packets that are not ECN-capable.

## 6. Protocol Ossification Considerations

Accurate loss information is not critical to the operation of any protocol, though its presence for a sufficient number of flows is important for the operation of networks.

The loss bits are amenable to "greasing" described in [RFC8701], if the protocol designers are not ready to dedicate (and ossify) bits used for loss reporting to this function. The greasing could be accomplished similarly to the Latency Spin bit greasing in [QUIC-TRANSPORT]. Namely, implementations could decide that a fraction of flows should not encode loss information in the loss bits and, instead, the bits would be set to arbitrary values. The observers would need to be ready to ignore flows with loss information more resembling noise than the expected signal.

## 7. Security Considerations

Passive loss observation has been a part of the network operations for a long time, so exposing loss information to the network does not add new security concerns for protocols that are currently observable.

In the absence of upstream packet loss, the Q bit signal does not provide any information that cannot be observed by simply counting packets transiting a network path. In the presence of upstream packet loss, the Q bit will disclose the loss, but this is information about the environment and not the endpoint state. The L bit signal discloses internal state of the protocol's loss detection machinery, but this state can often be gleamed by timing packets and observing congestion controller response. Hence, loss bits do not provide a viable new mechanism to attack data integrity and secrecy.

### 7.1. Optimistic ACK Attack

A defense against an Optimistic ACK Attack, described in [QUIC-TRANSPORT], involves a sender randomly skipping packet numbers to detect a receiver acknowledging packet numbers that have never been received. The Q bit signal may inform the attacker which packet numbers were skipped on purpose and which had been actually lost (and are, therefore, safe for the attacker to acknowledge). To use the Q bit for this purpose, the attacker must first receive at least an entire Q Run of packets, which renders the attack ineffective against a delay-sensitive congestion controller.

A protocol that is more susceptible to an Optimistic ACK Attack with the loss signal provided by Q bit and uses a loss-based congestion controller, SHOULD shorten the current Q Run by the number of skipped packets numbers. For example, skipping a single packet number will invert the sQuare signal one outgoing packet sooner.

## 8. Privacy Considerations

To minimize unintentional exposure of information, loss bits provide an explicit loss signal - a preferred way to share information per [RFC8558].

New protocols commonly have specific privacy goals, and loss reporting must ensure that loss information does not compromise those privacy goals. For example, [QUIC-TRANSPORT] allows changing Connection IDs in the middle of a connection to reduce the likelihood of a passive observer linking old and new subflows to the same device. A QUIC implementation would need to reset all counters when it changes the destination (IP address or UDP port) or the Connection ID used for outgoing packets. It would also need to avoid incrementing Unreported Loss counter for loss of packets sent to a different destination or with a different Connection ID.

## 9. IANA Considerations

This document makes no request of IANA.

## 10. Change Log

### 10.1. Since version 02

- Minor improvement and clarifications

## 10.2. Since version 01

- Clarified Q Period selection
- Added an optional E (ECN-Echo Event) bit
- Clarified L bit calculation for protocols that allow partial data loss due to a change in segmentation (such as TCP)

## 10.3. Since version 00

- Addressed review comments
- Improved guidelines for privacy protections for QIUC

## 11. Acknowledgments

The sSquare bit was originally suggested by Kazuho Oku in early proposals for loss measurement and is an instance of the "alternate marking" as defined in [RFC8321].

## 12. References

## 12.1. Normative References

- [ConEx] Mathis, M. and B. Briscoe, "Congestion Exposure (ConEx) Concepts, Abstract Mechanism, and Requirements", RFC 7713, DOI 10.17487/RFC7713, December 2015, <<https://www.rfc-editor.org/info/rfc7713>>.
- [ConEx-TCP] Kuehlewind, M., Ed. and R. Scheffenegger, "TCP Modifications for Congestion Exposure (ConEx)", RFC 7786, DOI 10.17487/RFC7786, May 2016, <<https://www.rfc-editor.org/info/rfc7786>>.
- [ECN] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <<https://www.rfc-editor.org/info/rfc3168>>.
- [IP] Postel, J., "Internet Protocol", STD 5, RFC 791, DOI 10.17487/RFC0791, September 1981, <<https://www.rfc-editor.org/info/rfc791>>.

- [IPv6] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.
- [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>>.
- [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>>.
- [RFC8558] Hardie, T., Ed., "Transport Protocol Path Signals", RFC 8558, DOI 10.17487/RFC8558, April 2019, <<https://www.rfc-editor.org/info/rfc8558>>.
- [TCP] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <<https://www.rfc-editor.org/info/rfc793>>.

## 12.2. Informative References

- [ACCURATE] Briscoe, B., Kuehlewind, M., and R. Scheffenegger, "More Accurate ECN Feedback in TCP", draft-ietf-tcpm-accurate-ecn-11 (work in progress), March 2020.
- [IPv6AltMark] Fioccola, G., Zhou, T., Cociglio, M., Qin, F., and R. Pang, "IPv6 Application of the Alternate Marking Method", draft-ietf-6man-ipv6-alt-mark-01 (work in progress), June 2020.
- [QUIC-TRANSPORT] Iyengar, J. and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport", draft-ietf-quic-transport-29 (work in progress), June 2020.
- [RFC8517] Dolson, D., Ed., Snellman, J., Boucadair, M., Ed., and C. Jacquenet, "An Inventory of Transport-Centric Functions Provided by Middleboxes: An Operator Perspective", RFC 8517, DOI 10.17487/RFC8517, February 2019, <<https://www.rfc-editor.org/info/rfc8517>>.

[RFC8701] Benjamin, D., "Applying Generate Random Extensions And Sustain Extensibility (GREASE) to TLS Extensibility", RFC 8701, DOI 10.17487/RFC8701, January 2020, <<https://www.rfc-editor.org/info/rfc8701>>.

[TRANSPORT-ENCRYPT] Fairhurst, G. and C. Perkins, "Considerations around Transport Header Confidentiality, Network Operations, and the Evolution of Internet Transport Protocols", draft-ietf-tsvwg-transport-encrypt-16 (work in progress), July 2020.

[UDP-OPTIONS] Touch, J., "Transport Options for UDP", draft-ietf-tsvwg-udp-options-08 (work in progress), September 2019.

[UDP-SURPLUS] Herbert, T., "UDP Surplus Header", draft-herbert-udp-space-hdr-01 (work in progress), July 2019.

#### Authors' Addresses

Alexandre Ferrieux (editor)  
Orange Labs

EMail: [alexandre.ferrieux@orange.com](mailto:alexandre.ferrieux@orange.com)

Isabelle Hamchaoui (editor)  
Orange Labs

EMail: [isabelle.hamchaoui@orange.com](mailto:isabelle.hamchaoui@orange.com)

Igor Lubashev (editor)  
Akamai Technologies

EMail: [ilubashe@akamai.com](mailto:ilubashe@akamai.com)

Dmitri Tikhonov (editor)  
LiteSpeed Technologies

EMail: [dtikhonov@litespeedtech.com](mailto:dtikhonov@litespeedtech.com)



Internet Engineering Task Force  
Internet-Draft  
Intended status: Standards Track  
Expires: September 25, 2019

S. Cheshire  
Apple Inc.  
March 24, 2019

Discovery Proxy for Multicast DNS-Based Service Discovery  
draft-ietf-dnssd-hybrid-10

Abstract

This document specifies a network proxy that uses Multicast DNS to automatically populate the wide-area unicast Domain Name System namespace with records describing devices and services found on the local link.

Status of This Memo

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

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

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

This Internet-Draft will expire on September 25, 2019.

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 (<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. Operational Analogy . . . . .	6
3. Conventions and Terminology Used in this Document . . . . .	7
4. Compatibility Considerations . . . . .	7
5. Discovery Proxy Operation . . . . .	8
5.1. Delegated Subdomain for Service Discovery Records . . . . .	9
5.2. Domain Enumeration . . . . .	11
5.2.1. Domain Enumeration via Unicast Queries . . . . .	11
5.2.2. Domain Enumeration via Multicast Queries . . . . .	13
5.3. Delegated Subdomain for LDH Host Names . . . . .	14
5.4. Delegated Subdomain for Reverse Mapping . . . . .	16
5.5. Data Translation . . . . .	18
5.5.1. DNS TTL limiting . . . . .	18
5.5.2. Suppressing Unusable Records . . . . .	19
5.5.3. NSEC and NSEC3 queries . . . . .	20
5.5.4. No Text Encoding Translation . . . . .	20
5.5.5. Application-Specific Data Translation . . . . .	21
5.6. Answer Aggregation . . . . .	23
6. Administrative DNS Records . . . . .	27
6.1. DNS SOA (Start of Authority) Record . . . . .	27
6.2. DNS NS Records . . . . .	28
6.3. DNS Delegation Records . . . . .	28
6.4. DNS SRV Records . . . . .	29
7. DNSSEC Considerations . . . . .	30
7.1. On-line signing only . . . . .	30
7.2. NSEC and NSEC3 Records . . . . .	30
8. IPv6 Considerations . . . . .	31
9. Security Considerations . . . . .	32
9.1. Authenticity . . . . .	32
9.2. Privacy . . . . .	32
9.3. Denial of Service . . . . .	32
10. IANA Considerations . . . . .	33
11. Acknowledgments . . . . .	33
12. References . . . . .	34
12.1. Normative References . . . . .	34
12.2. Informative References . . . . .	35
Appendix A. Implementation Status . . . . .	38
A.1. Already Implemented and Deployed . . . . .	38
A.2. Already Implemented . . . . .	38
A.3. Partially Implemented . . . . .	39
Author's Address . . . . .	39

## 1. Introduction

Multicast DNS [RFC6762] and its companion technology DNS-based Service Discovery [RFC6763] were created to provide IP networking with the ease-of-use and autoconfiguration for which AppleTalk was well known [RFC6760] [ZC] [Roadmap].

For a small home network consisting of just a single link (or a few physical links bridged together to appear as a single logical link from the point of view of IP) Multicast DNS [RFC6762] is sufficient for client devices to look up the ".local" host names of peers on the same home network, and to use Multicast DNS-Based Service Discovery (DNS-SD) [RFC6763] to discover services offered on that home network.

For a larger network consisting of multiple links that are interconnected using IP-layer routing instead of link-layer bridging, link-local Multicast DNS alone is insufficient because link-local Multicast DNS packets, by design, are not propagated onto other links.

Using link-local multicast packets for Multicast DNS was a conscious design choice [RFC6762]. Even when limited to a single link, multicast traffic is still generally considered to be more expensive than unicast, because multicast traffic impacts many devices, instead of just a single recipient. In addition, with some technologies like Wi-Fi [IEEE-11], multicast traffic is inherently less efficient and less reliable than unicast, because Wi-Fi multicast traffic is sent at lower data rates, and is not acknowledged [Mcast]. Increasing the amount of expensive multicast traffic by flooding it across multiple links would make the traffic load even worse.

Partitioning the network into many small links curtails the spread of expensive multicast traffic, but limits the discoverability of services. At the opposite end of the spectrum, using a very large local link with thousands of hosts enables better service discovery, but at the cost of larger amounts of multicast traffic.

Performing DNS-Based Service Discovery using purely Unicast DNS is more efficient and doesn't require large multicast domains, but does require that the relevant data be available in the Unicast DNS namespace. The Unicast DNS namespace in question could fall within a traditionally assigned globally unique domain name, or could use a private local unicast domain name such as ".home.arpa" [RFC8375].

In the DNS-SD specification [RFC6763], Section 10 ("Populating the DNS with Information") discusses various possible ways that a service's PTR, SRV, TXT and address records can make their way into the Unicast DNS namespace, including manual zone file configuration

[RFC1034] [RFC1035], DNS Update [RFC2136] [RFC3007] and proxies of various kinds.

Making the relevant data available in the Unicast DNS namespace by manual DNS configuration is one option. This option has been used for many years at IETF meetings to advertise the IETF Terminal Room printer. Details of this example are given in Appendix A of the Roadmap document [Roadmap]. However, this manual DNS configuration is labor intensive, error prone, and requires a reasonable degree of DNS expertise.

Populating the Unicast DNS namespace via DNS Update by the devices offering the services themselves is another option [RegProt] [DNS-UL]. However, this requires configuration of DNS Update keys on those devices, which has proven onerous and impractical for simple devices like printers and network cameras.

Hence, to facilitate efficient and reliable DNS-Based Service Discovery, a compromise is needed that combines the ease-of-use of Multicast DNS with the efficiency and scalability of Unicast DNS.

This document specifies a type of proxy called a "Discovery Proxy" that uses Multicast DNS [RFC6762] to discover Multicast DNS records on its local link, and makes corresponding DNS records visible in the Unicast DNS namespace.

In principle, similar mechanisms could be defined using other local service discovery protocols, to discover local information and then make corresponding DNS records visible in the Unicast DNS namespace. Such mechanisms for other local service discovery protocols could be addressed in future documents.

The design of the Discovery Proxy is guided by the previously published requirements document [RFC7558].

In simple terms, a descriptive DNS name is chosen for each link in an organization. Using a DNS NS record, responsibility for that DNS name is delegated to a Discovery Proxy physically attached to that link. Now, when a remote client issues a unicast query for a name falling within the delegated subdomain, the normal DNS delegation mechanism results in the unicast query arriving at the Discovery Proxy, since it has been declared authoritative for those names. Now, instead of consulting a textual zone file on disk to discover the answer to the query, as a traditional DNS server would, a Discovery Proxy consults its local link, using Multicast DNS, to find the answer to the question.

For fault tolerance reasons there may be more than one Discovery Proxy serving a given link.

Note that the Discovery Proxy uses a "pull" model. The local link is not queried using Multicast DNS until some remote client has requested that data. In the idle state, in the absence of client requests, the Discovery Proxy sends no packets and imposes no burden on the network. It operates purely "on demand".

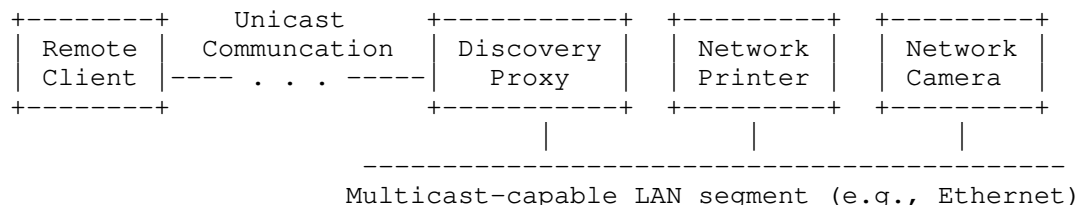
An alternative proposal that has been discussed is a proxy that performs DNS updates to a remote DNS server on behalf of the Multicast DNS devices on the local network. The difficulty with this is that Multicast DNS devices do not routinely announce their records on the network. Generally they remain silent until queried. This means that the complete set of Multicast DNS records in use on a link can only be discovered by active querying, not by passive listening. Because of this, a proxy can only know what names exist on a link by issuing queries for them, and since it would be impractical to issue queries for every possible name just to find out which names exist and which do not, there is no reasonable way for a proxy to programmatically learn all the answers it would need to push up to the remote DNS server using DNS Update. Even if such a mechanism were possible, it would risk generating high load on the network continuously, even when there are no clients with any interest in that data.

Hence, having a model where the query comes to the Discovery Proxy is much more efficient than a model where the Discovery Proxy pushes the answers out to some other remote DNS server.

A client seeking to discover services and other information achieves this by sending traditional DNS queries to the Discovery Proxy, or by sending DNS Push Notification subscription requests [Push].

How a client discovers what domain name(s) to use for its service discovery queries, (and consequently what Discovery Proxy or Proxies to use) is described in Section 5.2.

The diagram below illustrates a network topology using a Discovery Proxy to provide discovery service to a remote client.



## 2. Operational Analogy

A Discovery Proxy does not operate as a multicast relay, or multicast forwarder. There is no danger of multicast forwarding loops that result in traffic storms, because no multicast packets are forwarded. A Discovery Proxy operates as a *\*proxy\** for a remote client, performing queries on its behalf and reporting the results back.

A reasonable analogy is making a telephone call to a colleague at your workplace and saying, "I'm out of the office right now. Would you mind bringing up a printer browser window and telling me the names of the printers you see?" That entails no risk of a forwarding loop causing a traffic storm, because no multicast packets are sent over the telephone call.

A similar analogy, instead of enlisting another human being to initiate the service discovery operation on your behalf, is to log into your own desktop work computer using screen sharing, and then run the printer browser yourself to see the list of printers. Or log in using ssh and type "dns-sd -B \_ipp.\_tcp" and observe the list of discovered printer names. In neither case is there any risk of a forwarding loop causing a traffic storm, because no multicast packets are being sent over the screen sharing or ssh connection.

The Discovery Proxy provides another way of performing remote queries, except using a different protocol instead of screen sharing or ssh.

When the Discovery Proxy software performs Multicast DNS operations, the exact same Multicast DNS caching mechanisms are applied as when any other client software on that Discovery Proxy device performs Multicast DNS operations, whether that be running a printer browser client locally, or a remote user running the printer browser client via a screen sharing connection, or a remote user logged in via ssh running a command-line tool like "dns-sd", or a remote user sending DNS requests that cause a Discovery Proxy to perform discovery operations on its behalf.

### 3. Conventions and Terminology Used in this Document

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 "Key words for use in RFCs to Indicate Requirement Levels", when, and only when, they appear in all capitals, as shown here [RFC2119] [RFC8174].

The Discovery Proxy builds on Multicast DNS, which works between hosts on the same link. For the purposes of this document a set of hosts is considered to be "on the same link" if:

- o when any host from that set sends a packet to any other host in that set, using unicast, multicast, or broadcast, the entire link-layer packet payload arrives unmodified, and
- o a broadcast sent over that link, by any host from that set of hosts, can be received by every other host in that set.

The link-layer *\*header\** may be modified, such as in Token Ring Source Routing [IEEE-5], but not the link-layer *\*payload\**. In particular, if any device forwarding a packet modifies any part of the IP header or IP payload then the packet is no longer considered to be on the same link. This means that the packet may pass through devices such as repeaters, bridges, hubs or switches and still be considered to be on the same link for the purpose of this document, but not through a device such as an IP router that decrements the IP TTL or otherwise modifies the IP header.

### 4. Compatibility Considerations

No changes to existing devices are required to work with a Discovery Proxy.

Existing devices that advertise services using Multicast DNS work with Discovery Proxy.

Existing clients that support DNS-Based Service Discovery over Unicast DNS work with Discovery Proxy. Service Discovery over Unicast DNS was introduced in Mac OS X 10.4 in April 2005, as is included in Apple products introduced since then, including iPhone and iPad, as well as products from other vendors, such as Microsoft Windows 10.

An overview of the larger collection of related Service Discovery technologies, and how Discovery Proxy relates to those, is given in the Service Discovery Road Map document [Roadmap].

## 5. Discovery Proxy Operation

In a typical configuration, a Discovery Proxy is configured to be authoritative [RFC1034] [RFC1035] for four or more DNS subdomains, and authority for these subdomains is delegated to it via NS records:

A DNS subdomain for service discovery records.

This subdomain name may contain rich text, including spaces and other punctuation. This is because this subdomain name is used only in graphical user interfaces, where rich text is appropriate.

A DNS subdomain for host name records.

This subdomain name SHOULD be limited to letters, digits and hyphens, to facilitate convenient use of host names in command-line interfaces.

One or more DNS subdomains for IPv4 Reverse Mapping records.

These subdomains will have names that ends in "in-addr.arpa."

One or more DNS subdomains for IPv6 Reverse Mapping records.

These subdomains will have names that ends in "ip6.arpa."

In an enterprise network the naming and delegation of these subdomains is typically performed by conscious action of the network administrator. In a home network naming and delegation would typically be performed using some automatic configuration mechanism such as HNCP [RFC7788].

These three varieties of delegated subdomains (service discovery, host names, and reverse mapping) are described below in Section 5.1, Section 5.3 and Section 5.4.

How a client discovers where to issue its service discovery queries is described below in Section 5.2.



### 5.1. Delegated Subdomain for Service Discovery Records

In its simplest form, each link in an organization is assigned a unique Unicast DNS domain name, such as "Building 1.example.com" or "2nd Floor.Building 3.example.com". Grouping multiple links under a single Unicast DNS domain name is to be specified in a future companion document, but for the purposes of this document, assume that each link has its own unique Unicast DNS domain name. In a graphical user interface these names are not displayed as strings with dots as shown above, but something more akin to a typical file browser graphical user interface (which is harder to illustrate in a text-only document) showing folders, subfolders and files in a file system.

*example.com*	Building 1	1st Floor	Alice's printer
	Building 2	*2nd Floor*	Bob's printer
	*Building 3*	3rd Floor	Charlie's printer
	Building 4	4th Floor	
	Building 5		
	Building 6		

Figure 1: Illustrative GUI

Each named link in an organization has one or more Discovery Proxies which serve it. This Discovery Proxy function for each link could be performed by a device like a router or switch that is physically attached to that link. In the parent domain, NS records are used to delegate ownership of each defined link name

(e.g., "Building 1.example.com") to the one or more Discovery Proxies that serve the named link. In other words, the Discovery Proxies are the authoritative name servers for that subdomain. As in the rest of DNS-Based Service Discovery, all names are represented as-is using plain UTF-8 encoding, and, as described in Section 5.5.4, no text encoding translations are performed.

With appropriate VLAN configuration [IEEE-1Q] a single Discovery Proxy device could have a logical presence on many links, and serve as the Discovery Proxy for all those links. In such a configuration the Discovery Proxy device would have a single physical Ethernet [IEEE-3] port, configured as a VLAN trunk port, which would appear to software on that device as multiple virtual Ethernet interfaces, one connected to each of the VLAN links.

As an alternative to using VLAN technology, using a Multicast DNS Discovery Relay [Relay] is another way that a Discovery Proxy can have a 'virtual' presence on a remote link.

When a DNS-SD client issues a Unicast DNS query to discover services in a particular Unicast DNS subdomain (e.g., "\_printer.\_tcp.Building 1.example.com. PTR ?") the normal DNS delegation mechanism results in that query being forwarded until it reaches the delegated authoritative name server for that subdomain, namely the Discovery Proxy on the link in question. Like a conventional Unicast DNS server, a Discovery Proxy implements the usual Unicast DNS protocol [RFC1034] [RFC1035] over UDP and TCP. However, unlike a conventional Unicast DNS server that generates answers from the data in its manually-configured zone file, a Discovery Proxy generates answers using Multicast DNS. A Discovery Proxy does this by consulting its Multicast DNS cache and/or issuing Multicast DNS queries, as appropriate, according to the usual protocol rules of Multicast DNS [RFC6762], for the corresponding Multicast DNS name, type and class, with the delegated zone part of the name replaced with ".local" (e.g., in this case, "\_printer.\_tcp.local. PTR ?"). Then, from the received Multicast DNS data, the Discovery Proxy synthesizes the appropriate Unicast DNS response, with the ".local" top-level label replaced with the name of the delegated zone. How long the Discovery Proxy should wait to accumulate Multicast DNS responses before sending its unicast reply is described below in Section 5.6.

The existing Multicast DNS caching mechanism is used to minimize unnecessary Multicast DNS queries on the wire. The Discovery Proxy is acting as a client of the underlying Multicast DNS subsystem, and benefits from the same caching and efficiency measures as any other client using that subsystem.

Note that the contents of the delegated zone, generated as it is by performing ".local" Multicast DNS queries, mirrors the records available on the local link via Multicast DNS very closely, but not precisely. There is not a full bidirectional equivalence between the two. Certain records that are available via Multicast DNS may not have equivalents in the delegated zone, possibly because they are invalid or not relevant in the delegated zone, or because they are being suppressed because they are unusable outside the local link (see Section 5.5.2). Conversely, certain records that appear in the delegated zone may not have corresponding records available on the local link via Multicast DNS. In particular there are certain administrative SRV records (see Section 6) that logically fall within the delegated zone, but semantically represent metadata *about* the zone rather than records *within* the zone, and consequently these administrative records in the delegated zone do not have any corresponding counterparts in the Multicast DNS namespace of the local link.

## 5.2. Domain Enumeration

A DNS-SD client performs Domain Enumeration [RFC6763] via certain PTR queries, using both unicast and multicast. If it receives a Domain Name configuration via DHCP option 15 [RFC2132], then it issues unicast queries using this domain. It issues unicast queries using names derived from its IPv4 subnet address(es) and IPv6 prefix(es). These are described below in Section 5.2.1. It also issues multicast Domain Enumeration queries in the "local" domain [RFC6762]. These are described below in Section 5.2.2. The results of all the Domain Enumeration queries are combined for Service Discovery purposes.

### 5.2.1. Domain Enumeration via Unicast Queries

The administrator creates Domain Enumeration PTR records [RFC6763] to inform clients of available service discovery domains. Two varieties of such Domain Enumeration PTR records exist; those with names derived from the domain name communicated to the clients via DHCP, and those with names derived from IPv4 subnet address(es) and IPv6 prefix(es) in use by the clients. Below is an example showing the name-based variety:

b._dns-sd._udp.example.com.	PTR	Building 1.example.com.
	PTR	Building 2.example.com.
	PTR	Building 3.example.com.
	PTR	Building 4.example.com.
db._dns-sd._udp.example.com.	PTR	Building 1.example.com.
lb._dns-sd._udp.example.com.	PTR	Building 1.example.com.

The meaning of these records is defined in the DNS Service Discovery specification [RFC6763] but for convenience is repeated here. The "b" ("browse") records tell the client device the list of browsing domains to display for the user to select from. The "db" ("default browse") record tells the client device which domain in that list should be selected by default. The "db" domain MUST be one of the domains in the "b" list; if not then no domain is selected by default. The "lb" ("legacy browse") record tells the client device which domain to automatically browse on behalf of applications that don't implement UI for multi-domain browsing (which is most of them, at the time of writing). The "lb" domain is often the same as the "db" domain, or sometimes the "db" domain plus one or more others that should be included in the list of automatic browsing domains for legacy clients.

Note that in the example above, for clarity, space characters in names are shown as actual spaces. If this data is manually entered

into a textual zone file for authoritative server software such as BIND, care must be taken because the space character is used as a field separator, and other characters like dot ('.'), semicolon (';'), dollar ('\$'), backslash ('\'), etc., also have special meaning. These characters have to be escaped when entered into a textual zone file, following the rules in Section 5.1 of the DNS specification [RFC1035]. For example, a literal space in a name is represented in the textual zone file using '\032', so "Building 1.example.com." is entered as "Building\0321.example.com."

DNS responses are limited to a maximum size of 65535 bytes. This limits the maximum number of domains that can be returned for a Domain Enumeration query, as follows:

A DNS response header is 12 bytes. That's typically followed by a single qname (up to 256 bytes) plus qtype (2 bytes) and qclass (2 bytes), leaving 65275 for the Answer Section.

An Answer Section Resource Record consists of:

- o Owner name, encoded as a two-byte compression pointer
- o Two-byte rrtype (type PTR)
- o Two-byte rrclass (class IN)
- o Four-byte ttl
- o Two-byte rdlength
- o rdata (domain name, up to 256 bytes)

This means that each Resource Record in the Answer Section can take up to 268 bytes total, which means that the Answer Section can contain, in the worst case, no more than 243 domains.

In a more typical scenario, where the domain names are not all maximum-sized names, and there is some similarity between names so that reasonable name compression is possible, each Answer Section Resource Record may average 140 bytes, which means that the Answer Section can contain up to 466 domains.

It is anticipated that this should be sufficient for even a large corporate network or university campus.

### 5.2.2. Domain Enumeration via Multicast Queries

In the case where Discovery Proxy functionality is widely deployed within an enterprise (either by having a Discovery Proxy on each link, or by having a Discovery Proxy with a remote 'virtual' presence on each link using VLANs or Multicast DNS Discovery Relays [Relay]) this offers an additional way to provide Domain Enumeration data for clients.

A Discovery Proxy can be configured to generate Multicast DNS responses for the following Multicast DNS Domain Enumeration queries issued by clients:

b._dns-sd._udp.local.	PTR	?
db._dns-sd._udp.local.	PTR	?
lb._dns-sd._udp.local.	PTR	?

This provides the ability for Discovery Proxies to indicate recommended browsing domains to DNS-SD clients on a per-link granularity. In some enterprises it may be preferable to provide this per-link configuration data in the form of Discovery Proxy configuration, rather than populating the Unicast DNS servers with the same data (in the "ip6.arpa" or "in-addr.arpa" domains).

Regardless of how the network operator chooses to provide this configuration data, clients will perform Domain Enumeration via both unicast and multicast queries, and then combine the results of these queries.

### 5.3. Delegated Subdomain for LDH Host Names

DNS-SD service instance names and domains are allowed to contain arbitrary Net-Unicode text [RFC5198], encoded as precomposed UTF-8 [RFC3629].

Users typically interact with service discovery software by viewing a list of discovered service instance names on a display, and selecting one of them by pointing, touching, or clicking. Similarly, in software that provides a multi-domain DNS-SD user interface, users view a list of offered domains on the display and select one of them by pointing, touching, or clicking. To use a service, users don't have to remember domain or instance names, or type them; users just have to be able to recognize what they see on the display and touch or click on the thing they want.

In contrast, host names are often remembered and typed. Also, host names have historically been used in command-line interfaces where spaces can be inconvenient. For this reason, host names have traditionally been restricted to letters, digits and hyphens (LDH), with no spaces or other punctuation.

While we do want to allow rich text for DNS-SD service instance names and domains, it is advisable, for maximum compatibility with existing usage, to restrict host names to the traditional letter-digit-hyphen rules. This means that while a service name "My Printer.\_ipp.\_tcp.Building 1.example.com" is acceptable and desirable (it is displayed in a graphical user interface as an instance called "My Printer" in the domain "Building 1" at "example.com"), a host name "My-Printer.Building 1.example.com" is less desirable (because of the space in "Building 1").

To accomodate this difference in allowable characters, a Discovery Proxy SHOULD support having two separate subdomains delegated to it for each link it serves, one whose name is allowed to contain arbitrary Net-Unicode text [RFC5198], and a second more constrained subdomain whose name is restricted to contain only letters, digits, and hyphens, to be used for host name records (names of 'A' and 'AAAA' address records). The restricted names may be any valid name consisting of only letters, digits, and hyphens, including Punycode-encoded names [RFC3492].

For example, a Discovery Proxy could have the two subdomains "Building 1.example.com" and "bldg1.example.com" delegated to it. The Discovery Proxy would then translate these two Multicast DNS records:

```
My Printer._ipp._tcp.local. SRV 0 0 631 prnt.local.  
prnt.local.                A    203.0.113.2
```

into Unicast DNS records as follows:

```
My Printer._ipp._tcp.Building 1.example.com.  
                                SRV 0 0 631 prnt.bldg1.example.com.  
prnt.bldg1.example.com.        A    203.0.113.2
```

Note that the SRV record name is translated using the rich-text domain name ("Building 1.example.com") and the address record name is translated using the LDH domain ("bldg1.example.com").

A Discovery Proxy MAY support only a single rich text Net-Unicode domain, and use that domain for all records, including 'A' and 'AAAA' address records, but implementers choosing this option should be aware that this choice may produce host names that are awkward to use in command-line environments. Whether this is an issue depends on whether users in the target environment are expected to be using command-line interfaces.

A Discovery Proxy MUST NOT be restricted to support only a letter-digit-hyphen subdomain, because that results in an unnecessarily poor user experience.

As described above in Section 5.2.1, for clarity, space characters in names are shown as actual spaces. If this data were to be manually entered into a textual zone file (which it isn't) then spaces would need to be represented using '\032', so "My Printer.\_ipp.\_tcp.Building 1.example.com." would become "My\032Printer.\_ipp.\_tcp.Building\0321.example.com." Note that the '\032' representation does not appear in the network packets sent over the air. In the wire format of DNS messages, spaces are sent as spaces, not as '\032', and likewise, in a graphical user interface at the client device, spaces are shown as spaces, not as '\032'.

#### 5.4. Delegated Subdomain for Reverse Mapping

A Discovery Proxy can facilitate easier management of reverse mapping domains, particularly for IPv6 addresses where manual management may be more onerous than it is for IPv4 addresses.

To achieve this, in the parent domain, NS records are used to delegate ownership of the appropriate reverse mapping domain to the Discovery Proxy. In other words, the Discovery Proxy becomes the authoritative name server for the reverse mapping domain. For fault tolerance reasons there may be more than one Discovery Proxy serving a given link.

If a given link is using the IPv4 subnet 203.0.113/24, then the domain "113.0.203.in-addr.arpa" is delegated to the Discovery Proxy for that link.

For example, if a given link is using the IPv6 prefix 2001:0DB8:1234:5678/64, then the domain "8.7.6.5.4.3.2.1.8.b.d.0.1.0.0.2.ip6.arpa" is delegated to the Discovery Proxy for that link.

When a reverse mapping query arrives at the Discovery Proxy, it issues the identical query on its local link as a Multicast DNS query. The mechanism to force an apparently unicast name to be resolved using link-local Multicast DNS varies depending on the API set being used. For example, in the "dns\_sd.h" APIs (available on macOS, iOS, Bonjour for Windows, Linux and Android), using `kDNSServiceFlagsForceMulticast` indicates that the `DNSServiceQueryRecord()` call should perform the query using Multicast DNS. Other APIs sets have different ways of forcing multicast queries. When the host owning that IPv4 or IPv6 address responds with a name of the form "something.local", the Discovery Proxy rewrites that to use its configured LDH host name domain instead of "local", and returns the response to the caller.



For example, a Discovery Proxy with the two subdomains "113.0.203.in-addr.arpa" and "bldg1.example.com" delegated to it would translate this Multicast DNS record:

2.113.0.203.in-addr.arpa. PTR prnt.local.

into this Unicast DNS response:

2.113.0.203.in-addr.arpa. PTR prnt.bldg1.example.com.

Subsequent queries for the prnt.bldg1.example.com address record, falling as it does within the bldg1.example.com domain, which is delegated to the Discovery Proxy, will arrive at the Discovery Proxy, where they are answered by issuing Multicast DNS queries and using the received Multicast DNS answers to synthesize Unicast DNS responses, as described above.

Note that this design assumes that all addresses on a given IPv4 subnet or IPv6 prefix are mapped to hostnames using the Discovery Proxy mechanism. It would be possible to implement a Discovery Proxy that can be configured so that some address-to-name mappings are performed using Multicast DNS on the local link, while other address-to-name mappings within the same IPv4 subnet or IPv6 prefix are configured manually.

### 5.5. Data Translation

Generating the appropriate Multicast DNS queries involves, at the very least, translating from the configured DNS domain (e.g., "Building 1.example.com") on the Unicast DNS side to "local" on the Multicast DNS side.

Generating the appropriate Unicast DNS responses involves translating back from "local" to the appropriate configured DNS Unicast domain.

Other beneficial translation and filtering operations are described below.

#### 5.5.1. DNS TTL limiting

For efficiency, Multicast DNS typically uses moderately high DNS TTL values. For example, the typical TTL on DNS-SD PTR records is 75 minutes. What makes these moderately high TTLs acceptable is the cache coherency mechanisms built in to the Multicast DNS protocol which protect against stale data persisting for too long. When a service shuts down gracefully, it sends goodbye packets to remove its PTR records immediately from neighboring caches. If a service shuts down abruptly without sending goodbye packets, the Passive Observation Of Failures (POOF) mechanism described in Section 10.5 of the Multicast DNS specification [RFC6762] comes into play to purge the cache of stale data.

A traditional Unicast DNS client on a distant remote link does not get to participate in these Multicast DNS cache coherency mechanisms on the local link. For traditional Unicast DNS queries (those received without using Long-Lived Query [LLQ] or DNS Push Notification subscriptions [Push]) the DNS TTLs reported in the resulting Unicast DNS response MUST be capped to be no more than ten seconds.

Similarly, for negative responses, the negative caching TTL indicated in the SOA record [RFC2308] should also be ten seconds (Section 6.1).

This value of ten seconds is chosen based on user-experience considerations.

For negative caching, suppose a user is attempting to access a remote device (e.g., a printer), and they are unsuccessful because that device is powered off. Suppose they then place a telephone call and ask for the device to be powered on. We want the device to become available to the user within a reasonable time period. It is reasonable to expect it to take on the order of ten seconds for a simple device with a simple embedded operating system to power on.

Once the device is powered on and has announced its presence on the network via Multicast DNS, we would like it to take no more than a further ten seconds for stale negative cache entries to expire from Unicast DNS caches, making the device available to the user desiring to access it.

Similar reasoning applies to capping positive TTLs at ten seconds. In the event of a device moving location, getting a new DHCP address, or other renumbering events, we would like the updated information to be available to remote clients in a relatively timely fashion.

However, network administrators should be aware that many recursive (caching) DNS servers by default are configured to impose a minimum TTL of 30 seconds. If stale data appears to be persisting in the network to the extent that it adversely impacts user experience, network administrators are advised to check the configuration of their recursive DNS servers.

For received Unicast DNS queries that use LLQ [LLQ] or DNS Push Notifications [Push], the Multicast DNS record's TTL SHOULD be returned unmodified, because the Push Notification channel exists to inform the remote client as records come and go. For further details about Long-Lived Queries, and its newer replacement, DNS Push Notifications, see Section 5.6.

#### 5.5.2. Suppressing Unusable Records

A Discovery Proxy SHOULD offer a configurable option, enabled by default, to suppress Unicast DNS answers for records that are not useful outside the local link. When the option to suppress unusable records is enabled:

- o DNS A and AAAA records for IPv4 link-local addresses [RFC3927] and IPv6 link-local addresses [RFC4862] SHOULD be suppressed.
- o Similarly, for sites that have multiple private address realms [RFC1918], in cases where the Discovery Proxy can determine that the querying client is in a different address realm, private addresses SHOULD NOT be communicated to that client.
- o IPv6 Unique Local Addresses [RFC4193] SHOULD be suppressed in cases where the Discovery Proxy can determine that the querying client is in a different IPv6 address realm.
- o By the same logic, DNS SRV records that reference target host names that have no addresses usable by the requester should be suppressed, and likewise, DNS PTR records that point to unusable SRV records should be similarly be suppressed.

### 5.5.3. NSEC and NSEC3 queries

Multicast DNS devices do not routinely announce their records on the network. Generally they remain silent until queried. This means that the complete set of Multicast DNS records in use on a link can only be discovered by active querying, not by passive listening. Because of this, a Discovery Proxy can only know what names exist on a link by issuing queries for them, and since it would be impractical to issue queries for every possible name just to find out which names exist and which do not, a Discovery Proxy cannot programmatically generate the traditional NSEC [RFC4034] and NSEC3 [RFC5155] records which assert the nonexistence of a large range of names.

When queried for an NSEC or NSEC3 record type, the Discovery Proxy issues a qtype "ANY" query using Multicast DNS on the local link, and then generates an NSEC or NSEC3 response with a Type Bit Map signifying which record types do and do not exist for just the specific name queried, and no other names.

Multicast DNS NSEC records received on the local link MUST NOT be forwarded unmodified to a unicast querier, because there are slight differences in the NSEC record data. In particular, Multicast DNS NSEC records do not have the NSEC bit set in the Type Bit Map, whereas conventional Unicast DNS NSEC records do have the NSEC bit set.

### 5.5.4. No Text Encoding Translation

A Discovery Proxy does no translation between text encodings. Specifically, a Discovery Proxy does no translation between Punycode encoding [RFC3492] and UTF-8 encoding [RFC3629], either in the owner name of DNS records, or anywhere in the RDATA of DNS records (such as the RDATA of PTR records, SRV records, NS records, or other record types like TXT, where it is ambiguous whether the RDATA may contain DNS names). All bytes are treated as-is, with no attempt at text encoding translation. A client implementing DNS-based Service Discovery [RFC6763] will use UTF-8 encoding for its service discovery queries, which the Discovery Proxy passes through without any text encoding translation to the Multicast DNS subsystem. Responses from the Multicast DNS subsystem are similarly returned, without any text encoding translation, back to the requesting client.

#### 5.5.5. Application-Specific Data Translation

There may be cases where Application-Specific Data Translation is appropriate.

For example, AirPrint printers tend to advertise fairly verbose information about their capabilities in their DNS-SD TXT record. TXT record sizes in the range 500-1000 bytes are not uncommon. This information is a legacy from LPR printing, because LPR does not have in-band capability negotiation, so all of this information is conveyed using the DNS-SD TXT record instead. IPP printing does have in-band capability negotiation, but for convenience printers tend to include the same capability information in their IPP DNS-SD TXT records as well. For local mDNS use this extra TXT record information is inefficient, but not fatal. However, when a Discovery Proxy aggregates data from multiple printers on a link, and sends it via unicast (via UDP or TCP) this amount of unnecessary TXT record information can result in large responses. A DNS reply over TCP carrying information about 70 printers with an average of 700 bytes per printer adds up to about 50 kilobytes of data. Therefore, a Discovery Proxy that is aware of the specifics of an application-layer protocol such as AirPrint (which uses IPP) can elide unnecessary key/value pairs from the DNS-SD TXT record for better network efficiency.

Also, the DNS-SD TXT record for many printers contains an "adminurl" key something like "adminurl=http://printername.local/status.html". For this URL to be useful outside the local link, the embedded ".local" hostname needs to be translated to an appropriate name with larger scope. It is easy to translate ".local" names when they appear in well-defined places, either as a record's name, or in the rdata of record types like PTR and SRV. In the printing case, some application-specific knowledge about the semantics of the "adminurl" key is needed for the Discovery Proxy to know that it contains a name that needs to be translated. This is somewhat analogous to the need for NAT gateways to contain ALGs (Application-Specific Gateways) to facilitate the correct translation of protocols that embed addresses in unexpected places.

To avoid the need for application-specific knowledge about the semantics of particular TXT record keys, protocol designers are advised to avoid placing link-local names or link-local IP addresses in TXT record keys, if translation of those names or addresses would be required for off-link operation. In the printing case, the operational failure of failing to translate the "adminurl" key correctly is that, when accessed from a different link, printing will still work, but clicking the "Admin" UI button will fail to open the printer's administration page. Rather than duplicating the host name

from the service's SRV record in its "adminurl" key, thereby having the same host name appear in two places, a better design might have been to omit the host name from the "adminurl" key, and instead have the client implicitly substitute the target host name from the service's SRV record in place of a missing host name in the "adminurl" key. That way the desired host name only appears once, and it is in a well-defined place where software like the Discovery Proxy is expecting to find it.

Note that this kind of Application-Specific Data Translation is expected to be very rare. It is the exception, rather than the rule. This is an example of a common theme in computing. It is frequently the case that it is wise to start with a clean, layered design, with clear boundaries. Then, in certain special cases, those layer boundaries may be violated, where the performance and efficiency benefits outweigh the inelegance of the layer violation.

These layer violations are optional. They are done primarily for efficiency reasons, and generally should not be required for correct operation. A Discovery Proxy MAY operate solely at the mDNS layer, without any knowledge of semantics at the DNS-SD layer or above.

## 5.6. Answer Aggregation

In a simple analysis, simply gathering multicast answers and forwarding them in a unicast response seems adequate, but it raises the question of how long the Discovery Proxy should wait to be sure that it has received all the Multicast DNS answers it needs to form a complete Unicast DNS response. If it waits too little time, then it risks its Unicast DNS response being incomplete. If it waits too long, then it creates a poor user experience at the client end. In fact, there may be no time which is both short enough to produce a good user experience and at the same time long enough to reliably produce complete results.

Similarly, the Discovery Proxy -- the authoritative name server for the subdomain in question -- needs to decide what DNS TTL to report for these records. If the TTL is too long then the recursive (caching) name servers issuing queries on behalf of their clients risk caching stale data for too long. If the TTL is too short then the amount of network traffic will be more than necessary. In fact, there may be no TTL which is both short enough to avoid undesirable stale data and at the same time long enough to be efficient on the network.

Both these dilemmas are solved by use of DNS Long-Lived Queries (DNS LLQ) [LLQ] or its newer replacement, DNS Push Notifications [Push].

Clients supporting unicast DNS Service Discovery SHOULD implement DNS Push Notifications [Push] for improved user experience.

Clients and Discovery Proxies MAY support both DNS LLQ and DNS Push, and when talking to a Discovery Proxy that supports both, the client may use either protocol, as it chooses, though it is expected that only DNS Push will continue to be supported in the long run.

When a Discovery Proxy receives a query using DNS LLQ or DNS Push Notifications, it responds immediately using the Multicast DNS records it already has in its cache (if any). This provides a good client user experience by providing a near-instantaneous response. Simultaneously, the Discovery Proxy issues a Multicast DNS query on the local link to discover if there are any additional Multicast DNS records it did not already know about. Should additional Multicast DNS responses be received, these are then delivered to the client using additional DNS LLQ or DNS Push Notification update messages. The timeliness of such update messages is limited only by the timeliness of the device responding to the Multicast DNS query. If the Multicast DNS device responds quickly, then the update message is delivered quickly. If the Multicast DNS device responds slowly, then

the update message is delivered slowly. The benefit of using update messages is that the Discovery Proxy can respond promptly because it doesn't have to delay its unicast response to allow for the expected worst-case delay for receiving all the Multicast DNS responses. Even if a proxy were to try to provide reliability by assuming an excessively pessimistic worst-case time (thereby giving a very poor user experience) there would still be the risk of a slow Multicast DNS device taking even longer than that (e.g., a device that is not even powered on until ten seconds after the initial query is received) resulting in incomplete responses. Using update message solves this dilemma: even very late responses are not lost; they are delivered in subsequent update messages.

There are two factors that determine specifically how responses are generated:

The first factor is whether the query from the client used LLQ or DNS Push Notifications (used for long-lived service browsing PTR queries) or not (used for one-shot operations like SRV or address record queries). Note that queries using LLQ or DNS Push Notifications are received directly from the client. Queries not using LLQ or DNS Push Notifications are generally received via the client's configured recursive (caching) name server.

The second factor is whether the Discovery Proxy already has at least one record in its cache that positively answers the question.

- o Not using LLQ or Push Notifications; no answer in cache:  
Issue an mDNS query, exactly as a local client would issue an mDNS query on the local link for the desired record name, type and class, including retransmissions, as appropriate, according to the established mDNS retransmission schedule [RFC6762]. As soon as any Multicast DNS response packet is received that contains one or more positive answers to that question (with or without the Cache Flush bit [RFC6762] set), or a negative answer (signified via a Multicast DNS NSEC record [RFC6762]), the Discovery Proxy generates a Unicast DNS response packet containing the corresponding (filtered and translated) answers and sends it to the remote client. If after six seconds no Multicast DNS answers have been received, cancel the mDNS query and return a negative response to the remote client. Six seconds is enough time to transmit three mDNS queries, and allow some time for responses to arrive.  
DNS TTLs in responses MUST be capped to at most ten seconds.  
(Reasoning: Queries not using LLQ or Push Notifications are generally queries that expect an answer from only one device, so the first response is also the only response.)



- o Not using LLQ or Push Notifications; at least one answer in cache:  
Send response right away to minimise delay.  
DNS TTLs in responses MUST be capped to at most ten seconds.  
No local mDNS queries are performed.  
(Reasoning: Queries not using LLQ or Push Notifications are generally queries that expect an answer from only one device. Given RRSset TTL harmonisation, if the proxy has one Multicast DNS answer in its cache, it can reasonably assume that it has all of them.)
- o Using LLQ or Push Notifications; no answer in cache:  
As in the case above with no answer in the cache, perform mDNS querying for six seconds, and send a response to the remote client as soon as any relevant mDNS response is received.  
If after six seconds no relevant mDNS response has been received, return negative response to the remote client (for LLQ; not applicable for Push Notifications).  
(Reasoning: We don't need to rush to send an empty answer.)  
Whether or not a relevant mDNS response is received within six seconds, the query remains active for as long as the client maintains the LLQ or Push Notification state, and if mDNS answers are received later, LLQ or Push Notification messages are sent.  
DNS TTLs in responses are returned unmodified.
- o Using LLQ or Push Notifications; at least one answer in cache:  
As in the case above with at least one answer in cache, send response right away to minimise delay.  
The query remains active for as long as the client maintains the LLQ or Push Notification state, and results in transmission of mDNS queries, with appropriate Known Answer lists, to determine if further answers are available. If additional mDNS answers are received later, LLQ or Push Notification messages are sent.  
(Reasoning: We want UI that is displayed very rapidly, yet continues to remain accurate even as the network environment changes.)  
DNS TTLs in responses are returned unmodified.

The "negative responses" referred to above are "no error no answer" negative responses, not NXDOMAIN. This is because the Discovery Proxy cannot know all the Multicast DNS domain names that may exist on a link at any given time, so any name with no answers may have child names that do exist, making it an "empty nonterminal" name.

Note that certain aspects of the behavior described here do not have to be implemented overtly by the Discovery Proxy; they occur naturally as a result of using existing Multicast DNS APIs.

For example, in the first case above (no LLQ or Push Notifications, and no answers in the cache) if a new Multicast DNS query is requested (either by a local client, or by the Discovery Proxy on behalf of a remote client), and there is not already an identical Multicast DNS query active, and there are no matching answers already in the Multicast DNS cache on the Discovery Proxy device, then this will cause a series of Multicast DNS query packets to be issued with exponential backoff. The exponential backoff sequence in some implementations starts at one second and then doubles for each retransmission (0, 1, 3, 7 seconds, etc.) and in others starts at one second and then triples for each retransmission (0, 1, 4, 13 seconds, etc.). In either case, if no response has been received after six seconds, that is long enough that the underlying Multicast DNS implementation will have sent three query packets without receiving any response. At that point the Discovery Proxy cancels its Multicast DNS query (so no further Multicast DNS query packets will be sent for this query) and returns a negative response to the remote client via unicast.

The six-second delay is chosen to be long enough to give enough time for devices to respond, yet short enough not to be too onerous for a human user waiting for a response. For example, using the "dig" DNS debugging tool, the current default settings result in it waiting a total of 15 seconds for a reply (three transmissions of the query packet, with a wait of 5 seconds after each packet) which is ample time for it to have received a negative reply from a Discovery Proxy after six seconds.

The statement that for a one-shot query (i.e., no LLQ or Push Notifications requested), if at least one answer is already available in the cache then a Discovery Proxy should not issue additional mDNS query packets, also occurs naturally as a result of using existing Multicast DNS APIs. If a new Multicast DNS query is requested (either locally, or by the Discovery Proxy on behalf of a remote client), for which there are relevant answers already in the Multicast DNS cache on the Discovery Proxy device, and after the answers are delivered the Multicast DNS query is then cancelled immediately, then no Multicast DNS query packets will be generated for this query.

## 6. Administrative DNS Records

### 6.1. DNS SOA (Start of Authority) Record

The MNAME field SHOULD contain the host name of the Discovery Proxy device (i.e., the same domain name as the rdata of the NS record delegating the relevant zone(s) to this Discovery Proxy device).

The RNAME field SHOULD contain the mailbox of the person responsible for administering this Discovery Proxy device.

The SERIAL field MUST be zero.

Zone transfers are undefined for Discovery Proxy zones, and consequently the REFRESH, RETRY and EXPIRE fields have no useful meaning for Discovery Proxy zones. These fields SHOULD contain reasonable default values. The RECOMMENDED values are: REFRESH 7200, RETRY 3600, EXPIRE 86400.

The MINIMUM field (used to control the lifetime of negative cache entries) SHOULD contain the value 10. The value of ten seconds is chosen based on user-experience considerations (see Section 5.5.1).

In the event that there are multiple Discovery Proxy devices on a link for fault tolerance reasons, this will result in clients receiving inconsistent SOA records (different MNAME, and possibly RNAME) depending on which Discovery Proxy answers their SOA query. However, since clients generally have no reason to use the MNAME or RNAME data, this is unlikely to cause any problems.

## 6.2. DNS NS Records

In the event that there are multiple Discovery Proxy devices on a link for fault tolerance reasons, the parent zone MUST be configured with NS records giving the names of all the Discovery Proxy devices on the link.

Each Discovery Proxy device MUST be configured to answer NS queries for the zone apex name by giving its own NS record, and the NS records of its fellow Discovery Proxy devices on the same link, so that it can return the correct answers for NS queries.

The target host name in the RDATA of an NS record MUST NOT reference a name that falls within any zone delegated to a Discovery Proxy. Apart from the zone apex name, all other host names that fall within a zone delegated to a Discovery Proxy correspond to local Multicast DNS host names, which logically belong to the respective Multicast DNS hosts defending those names, not the Discovery Proxy. Generally speaking, the Discovery Proxy does not own or control the delegated zone; it is merely a conduit to the corresponding ".local" namespace, which is controlled by the Multicast DNS hosts on that link. If an NS record were to reference a manually-determined host name that falls within a delegated zone, that manually-determined host name may inadvertently conflict with a corresponding ".local" host name that is owned and controlled by some device on that link.

## 6.3. DNS Delegation Records

Since the Multicast DNS specification [RFC6762] states that there can be no delegation (subdomains) within a ".local" namespace, this implies that any name within a zone delegated to a Discovery Proxy (except for the zone apex name itself) cannot have any answers for any DNS queries for RRTYPEs SOA, NS, or DS. Consequently:

- o for any query for the zone apex name of a zone delegated to a Discovery Proxy, the Discovery Proxy MUST generate the appropriate immediate answers as described above, and
- o for any query for RRTYPEs SOA, NS, or DS, for any name within a zone delegated to a Discovery Proxy, other than the zone apex name, instead of translating the query to its corresponding Multicast DNS ".local" equivalent, a Discovery Proxy MUST generate an immediate negative answer.

#### 6.4. DNS SRV Records

There are certain special DNS records that logically fall within the delegated unicast DNS subdomain, but rather than mapping to their corresponding ".local" namesakes, they actually contain metadata pertaining to the operation of the delegated unicast DNS subdomain itself. They do not exist in the corresponding ".local" namespace of the local link. For these queries a Discovery Proxy MUST generate immediate answers, whether positive or negative, to avoid delays while clients wait for their query to be answered. For example, if a Discovery Proxy does not implement Long-Lived Queries [LLQ] then it MUST return an immediate negative answer to tell the client this without delay, instead of passing the query through to the local network as a query for "\_dns-llq.\_udp.local.", and then waiting unsuccessfully for answers that will not be forthcoming.

If a Discovery Proxy implements Long-Lived Queries [LLQ] then it MUST positively respond to "\_dns-llq.\_udp.<zone> SRV" queries, "\_dns-llq.\_tcp.<zone> SRV" queries, and "\_dns-llq-tls.\_tcp.<zone> SRV" queries as appropriate, else it MUST return an immediate negative answer for those queries.

If a Discovery Proxy implements DNS Push Notifications [Push] then it MUST positively respond to "\_dns-push-tls.\_tcp.<zone>" queries, else it MUST return an immediate negative answer for those queries.

A Discovery Proxy MUST return an immediate negative answer for "\_dns-update.\_udp.<zone> SRV" queries, "\_dns-update.\_tcp.<zone> SRV" queries, and "\_dns-update-tls.\_tcp.<zone> SRV" queries, since using DNS Update [RFC2136] to change zones generated dynamically from local Multicast DNS data is not possible.

## 7. DNSSEC Considerations

### 7.1. On-line signing only

The Discovery Proxy acts as the authoritative name server for designated subdomains, and if DNSSEC is to be used, the Discovery Proxy needs to possess a copy of the signing keys, in order to generate authoritative signed data from the local Multicast DNS responses it receives. Off-line signing is not applicable to Discovery Proxy.

### 7.2. NSEC and NSEC3 Records

In DNSSEC NSEC [RFC4034] and NSEC3 [RFC5155] records are used to assert the nonexistence of certain names, also described as "authenticated denial of existence".

Since a Discovery Proxy only knows what names exist on the local link by issuing queries for them, and since it would be impractical to issue queries for every possible name just to find out which names exist and which do not, a Discovery Proxy cannot programmatically synthesize the traditional NSEC and NSEC3 records which assert the nonexistence of a large range of names. Instead, when generating a negative response, a Discovery Proxy programmatically synthesizes a single NSEC record assert the nonexistence of just the specific name queried, and no others. Since the Discovery Proxy has the zone signing key, it can do this on demand. Since the NSEC record asserts the nonexistence of only a single name, zone walking is not a concern, so NSEC3 is not necessary.

Note that this applies only to traditional immediate DNS queries, which may return immediate negative answers when no immediate positive answer is available. When used with a DNS Push Notification subscription [Push] there are no negative answers, merely the absence of answers so far, which may change in the future if answers become available.

## 8. IPv6 Considerations

An IPv4-only host and an IPv6-only host behave as "ships that pass in the night". Even if they are on the same Ethernet [IEEE-3], neither is aware of the other's traffic. For this reason, each link may have *\*two\** unrelated ".local." zones, one for IPv4 and one for IPv6. Since for practical purposes, a group of IPv4-only hosts and a group of IPv6-only hosts on the same Ethernet act as if they were on two entirely separate Ethernet segments, it is unsurprising that their use of the ".local." zone should occur exactly as it would if they really were on two entirely separate Ethernet segments.

It will be desirable to have a mechanism to 'stitch' together these two unrelated ".local." zones so that they appear as one. Such mechanism will need to be able to differentiate between a dual-stack (v4/v6) host participating in both ".local." zones, and two different hosts, one IPv4-only and the other IPv6-only, which are both trying to use the same name(s). Such a mechanism will be specified in a future companion document.

At present, it is RECOMMENDED that a Discovery Proxy be configured with a single domain name for both the IPv4 and IPv6 ".local." zones on the local link, and when a unicast query is received, it should issue Multicast DNS queries using both IPv4 and IPv6 on the local link, and then combine the results.

## 9. Security Considerations

### 9.1. Authenticity

A service proves its presence on a link by its ability to answer link-local multicast queries on that link. If greater security is desired, then the Discovery Proxy mechanism should not be used, and something with stronger security should be used instead, such as authenticated secure DNS Update [RFC2136] [RFC3007].

### 9.2. Privacy

The Domain Name System is, generally speaking, a global public database. Records that exist in the Domain Name System name hierarchy can be queried by name from, in principle, anywhere in the world. If services on a mobile device (like a laptop computer) are made visible via the Discovery Proxy mechanism, then when those services become visible in a domain such as "My House.example.com" that might indicate to (potentially hostile) observers that the mobile device is in my house. When those services disappear from "My House.example.com" that change could be used by observers to infer when the mobile device (and possibly its owner) may have left the house. The privacy of this information may be protected using techniques like firewalls, split-view DNS, and Virtual Private Networks (VPNs), as are customarily used today to protect the privacy of corporate DNS information.

The privacy issue is particularly serious for the IPv4 and IPv6 reverse zones. If the public delegation of the reverse zones points to the Discovery Proxy, and the Discovery Proxy is reachable globally, then it could leak a significant amount of information. Attackers could discover hosts that otherwise might not be easy to identify, and learn their hostnames. Attackers could also discover the existence of links where hosts frequently come and go.

The Discovery Proxy could also provide sensitive records only to authenticated users. This is a general DNS problem, not specific to the Discovery Proxy. Work is underway in the IETF to tackle this problem [RFC7626].

### 9.3. Denial of Service

A remote attacker could use a rapid series of unique Unicast DNS queries to induce a Discovery Proxy to generate a rapid series of corresponding Multicast DNS queries on one or more of its local links. Multicast traffic is generally more expensive than unicast traffic -- especially on Wi-Fi links -- which makes this attack particularly serious. To limit the damage that can be caused by such



attacks, a Discovery Proxy (or the underlying Multicast DNS subsystem which it utilizes) MUST implement Multicast DNS query rate limiting appropriate to the link technology in question. For today's 802.11b/g/n/ac Wi-Fi links (for which approximately 200 multicast packets per second is sufficient to consume approximately 100% of the wireless spectrum) a limit of 20 Multicast DNS query packets per second is RECOMMENDED. On other link technologies like Gigabit Ethernet higher limits may be appropriate. A consequence of this rate limiting is that a rogue remote client could issue an excessive number of queries, resulting in denial of service to other legitimate remote clients attempting to use that Discovery Proxy. However, this is preferable to a rogue remote client being able to inflict even greater harm on the local network, which could impact the correct operation of all local clients on that network.

#### 10. IANA Considerations

This document has no IANA Considerations.

#### 11. Acknowledgments

Thanks to Markus Stenberg for helping develop the policy regarding the four styles of unicast response according to what data is immediately available in the cache. Thanks to Anders Brandt, Ben Campbell, Tim Chown, Alissa Cooper, Spencer Dawkins, Ralph Droms, Joel Halpern, Ray Hunter, Joel Jaeggli, Warren Kumari, Ted Lemon, Alexey Melnikov, Kathleen Moriarty, Tom Pusateri, Eric Rescorla, Adam Roach, David Schinazi, Markus Stenberg, Dave Thaler, and Andrew Yourtchenko for their comments.

## 12. References

### 12.1. 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>>.
- [RFC1918] Rekhter, Y., Moskowitz, B., Karrenberg, D., de Groot, G., and E. Lear, "Address Allocation for Private Internets", BCP 5, RFC 1918, DOI 10.17487/RFC1918, February 1996, <<https://www.rfc-editor.org/info/rfc1918>>.
- [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>>.
- [RFC2308] Andrews, M., "Negative Caching of DNS Queries (DNS NCACHE)", RFC 2308, DOI 10.17487/RFC2308, March 1998, <<https://www.rfc-editor.org/info/rfc2308>>.
- [RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <<https://www.rfc-editor.org/info/rfc3629>>.
- [RFC3927] Cheshire, S., Aboba, B., and E. Guttman, "Dynamic Configuration of IPv4 Link-Local Addresses", RFC 3927, DOI 10.17487/RFC3927, May 2005, <<https://www.rfc-editor.org/info/rfc3927>>.
- [RFC4034] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "Resource Records for the DNS Security Extensions", RFC 4034, DOI 10.17487/RFC4034, March 2005, <<https://www.rfc-editor.org/info/rfc4034>>.
- [RFC4862] Thomson, S., Narten, T., and T. Jinmei, "IPv6 Stateless Address Autoconfiguration", RFC 4862, DOI 10.17487/RFC4862, September 2007, <<https://www.rfc-editor.org/info/rfc4862>>.

- [RFC5155] Laurie, B., Sisson, G., Arends, R., and D. Blacka, "DNS Security (DNSSEC) Hashed Authenticated Denial of Existence", RFC 5155, DOI 10.17487/RFC5155, March 2008, <<https://www.rfc-editor.org/info/rfc5155>>.
- [RFC5198] Klensin, J. and M. Padlipsky, "Unicode Format for Network Interchange", RFC 5198, DOI 10.17487/RFC5198, March 2008, <<https://www.rfc-editor.org/info/rfc5198>>.
- [RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <<https://www.rfc-editor.org/info/rfc6762>>.
- [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>>.
- [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>>.
- [RFC8490] Bellis, R., Cheshire, S., Dickinson, J., Dickinson, S., Lemon, T., and T. Pusateri, "DNS Stateful Operations", RFC 8490, DOI 10.17487/RFC8490, March 2019, <<https://www.rfc-editor.org/info/rfc8490>>.
- [Push] Pusateri, T. and S. Cheshire, "DNS Push Notifications", draft-ietf-dnssd-push-19 (work in progress), March 2019.

## 12.2. Informative References

- [Roadmap] Cheshire, S., "Service Discovery Road Map", draft-cheshire-dnssd-roadmap-03 (work in progress), October 2018.
- [DNS-UL] Sekar, K., "Dynamic DNS Update Leases", draft-sekar-dns-ul-01 (work in progress), August 2006.
- [LLQ] Cheshire, S. and M. Krochmal, "DNS Long-Lived Queries", draft-sekar-dns-llq-03 (work in progress), March 2019.
- [RegProt] Cheshire, S. and T. Lemon, "Service Registration Protocol for DNS-Based Service Discovery", draft-sctl-service-registration-00 (work in progress), July 2017.
- [Relay] Cheshire, S. and T. Lemon, "Multicast DNS Discovery Relay", draft-sctl-dnssd-mdns-relay-04 (work in progress), March 2018.

- [Mcast] Perkins, C., McBride, M., Stanley, D., Kumari, W., and J. Zuniga, "Multicast Considerations over IEEE 802 Wireless Media", draft-ietf-mboned-ieee802-mcast-problems-04 (work in progress), November 2018.
- [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>>.
- [RFC2136] Vixie, P., Ed., Thomson, S., Rekhter, Y., and J. Bound, "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, DOI 10.17487/RFC2136, April 1997, <<https://www.rfc-editor.org/info/rfc2136>>.
- [RFC3007] Wellington, B., "Secure Domain Name System (DNS) Dynamic Update", RFC 3007, DOI 10.17487/RFC3007, November 2000, <<https://www.rfc-editor.org/info/rfc3007>>.
- [RFC3492] Costello, A., "Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)", RFC 3492, DOI 10.17487/RFC3492, March 2003, <<https://www.rfc-editor.org/info/rfc3492>>.
- [RFC4193] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, DOI 10.17487/RFC4193, October 2005, <<https://www.rfc-editor.org/info/rfc4193>>.
- [RFC6760] Cheshire, S. and M. Krochmal, "Requirements for a Protocol to Replace the AppleTalk Name Binding Protocol (NBP)", RFC 6760, DOI 10.17487/RFC6760, February 2013, <<https://www.rfc-editor.org/info/rfc6760>>.
- [RFC7558] Lynn, K., Cheshire, S., Blanchet, M., and D. Migault, "Requirements for Scalable DNS-Based Service Discovery (DNS-SD) / Multicast DNS (mDNS) Extensions", RFC 7558, DOI 10.17487/RFC7558, July 2015, <<https://www.rfc-editor.org/info/rfc7558>>.
- [RFC7626] Bortzmeyer, S., "DNS Privacy Considerations", RFC 7626, DOI 10.17487/RFC7626, August 2015, <<https://www.rfc-editor.org/info/rfc7626>>.
- [RFC7788] Stenberg, M., Barth, S., and P. Pfister, "Home Networking Control Protocol", RFC 7788, DOI 10.17487/RFC7788, April 2016, <<https://www.rfc-editor.org/info/rfc7788>>.

- [RFC8375] Pfister, P. and T. Lemon, "Special-Use Domain 'home.arpa.'", RFC 8375, DOI 10.17487/RFC8375, May 2018, <<https://www.rfc-editor.org/info/rfc8375>>.
- [ohp] "Discovery Proxy (Hybrid Proxy) implementation for OpenWrt", <<https://github.com/sbyx/ohybridproxy/>>.
- [ZC] Cheshire, S. and D. Steinberg, "Zero Configuration Networking: The Definitive Guide", O'Reilly Media, Inc. , ISBN 0-596-10100-7, December 2005.
- [IEEE-1Q] "IEEE Standard for Local and metropolitan area networks -- Bridges and Bridged Networks", IEEE Std 802.1Q-2014, November 2014, <<http://standards.ieee.org/getieee802/download/802-1Q-2014.pdf>>.
- [IEEE-3] "Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications", IEEE Std 802.3-2008, December 2008, <<http://standards.ieee.org/getieee802/802.3.html>>.
- [IEEE-5] Institute of Electrical and Electronics Engineers, "Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 5: Token ring access method and physical layer specification", IEEE Std 802.5-1998, 1995.
- [IEEE-11] "Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications", IEEE Std 802.11-2007, June 2007, <<http://standards.ieee.org/getieee802/802.11.html>>.

## Appendix A. Implementation Status

Some aspects of the mechanism specified in this document already exist in deployed software. Some aspects are new. This section outlines which aspects already exist and which are new.

### A.1. Already Implemented and Deployed

Domain enumeration by the client (the "b.\_dns-sd.\_udp" queries) is already implemented and deployed.

Unicast queries to the indicated discovery domain is already implemented and deployed.

These are implemented and deployed in Mac OS X 10.4 and later (including all versions of Apple iOS, on all iPhone and iPads), in Bonjour for Windows, and in Android 4.1 "Jelly Bean" (API Level 16) and later.

Domain enumeration and unicast querying have been used for several years at IETF meetings to make Terminal Room printers discoverable from outside the Terminal room. When an IETF attendee presses Cmd-P on a Mac, or selects AirPrint on an iPad or iPhone, and the Terminal room printers appear, that is because the client is sending unicast DNS queries to the IETF DNS servers. A walk-through giving the details of this particular specific example is given in Appendix A of the Roadmap document [Roadmap].

### A.2. Already Implemented

A minimal portable Discovery Proxy implementation has been produced by Markus Stenberg and Steven Barth, which runs on OS X and several Linux variants including OpenWrt [ohp]. It was demonstrated at the Berlin IETF in July 2013.

Tom Pusateri has an implementation that runs on any Unix/Linux. It has a RESTful interface for management and an experimental demo CLI and web interface.

Ted Lemon also has produced a portable implementation of Discovery Proxy, which is available in the mDNSResponder open source code.

The Long-Lived Query mechanism [LLQ] referred to in this specification exists and is deployed, but was not standardized by the IETF. The IETF has developed a superior Long-Lived Query mechanism called DNS Push Notifications [Push], which is built on DNS Stateful Operations [RFC8490]. The pragmatic short-term deployment approach is for vendors to produce Discovery Proxies that implement both the

deployed Long-Lived Query mechanism [LLQ] (for today's clients) and the new DNS Push Notifications mechanism [Push] as the preferred long-term direction.

#### A.3. Partially Implemented

The current APIs make multiple domains visible to client software, but most client UI today lumps all discovered services into a single flat list. This is largely a chicken-and-egg problem. Application writers were naturally reluctant to spend time writing domain-aware UI code when few customers today would benefit from it. If Discovery Proxy deployment becomes common, then application writers will have a reason to provide better UI. Existing applications will work with the Discovery Proxy, but will show all services in a single flat list. Applications with improved UI will group services by domain.

#### Author's Address

Stuart Cheshire  
Apple Inc.  
One Apple Park Way  
Cupertino, California 95014  
USA

Phone: +1 (408) 996-1010  
Email: cheshire@apple.com

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: August 3, 2020

P. Pfister  
E. Vyncke  
Cisco  
T. Pauly  
Apple Inc.  
D. Schinazi  
Google LLC  
W. Shao  
Cisco  
January 31, 2020

Discovering Provisioning Domain Names and Data  
draft-ietf-intarea-provisioning-domains-11

Abstract

Provisioning Domains (PvDs) are defined as consistent sets of network configuration information. This allows hosts to manage connections to multiple networks and interfaces simultaneously, such as when a home router provides connectivity through both a broadband and cellular network provider.

This document defines a mechanism for explicitly identifying PvDs through a Router Advertisement (RA) option. This RA option announces a PvD identifier, which hosts can compare to differentiate between PvDs. The option can directly carry some information about a PvD and can optionally point to additional PvD information that can be retrieved using HTTP over TLS.

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 August 3, 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 . . . . .	3
1.1. Specification of Requirements . . . . .	4
2. Terminology . . . . .	4
3. Provisioning Domain Identification using Router Advertisements . . . . .	5
3.1. PvD ID Option for Router Advertisements . . . . .	5
3.2. Router Behavior . . . . .	8
3.3. Non-PvD-aware Host Behavior . . . . .	9
3.4. PvD-aware Host Behavior . . . . .	9
3.4.1. DHCPv6 configuration association . . . . .	10
3.4.2. DHCPv4 configuration association . . . . .	11
3.4.3. Connection Sharing by the Host . . . . .	11
3.4.4. Usage of DNS Servers . . . . .	12
4. Provisioning Domain Additional Information . . . . .	13
4.1. Retrieving the PvD Additional Information . . . . .	13
4.2. Operational Consideration to Providing the PvD Additional Information . . . . .	16
4.3. PvD Additional Information Format . . . . .	16
4.3.1. Example . . . . .	18
4.4. Detecting misconfiguration and misuse . . . . .	18
5. Operational Considerations . . . . .	19
5.1. Exposing Extra RA Options to PvD-Aware Hosts . . . . .	19
5.2. Different RAs for PvD-Aware and Non-PvD-Aware Hosts . . . . .	19
5.3. Enabling Multi-homing for PvD-Aware Hosts . . . . .	21
5.4. Providing Additional Information to PvD-Aware Hosts . . . . .	22
6. Security Considerations . . . . .	23
7. Privacy Considerations . . . . .	24
8. IANA Considerations . . . . .	25
8.1. New entry in the Well-Known URIs Registry . . . . .	26
8.2. Additional Information PvD Keys Registry . . . . .	26
8.3. PvD Option Flags Registry . . . . .	26

8.4. PvD JSON Media Type Registration . . . . .	27
9. Acknowledgments . . . . .	28
10. References . . . . .	28
10.1. Normative References . . . . .	28
10.2. Informative References . . . . .	30
Authors' Addresses . . . . .	32

## 1. Introduction

Provisioning Domains (PvDs) are defined in [RFC7556] as consistent sets of network configuration information. This information includes properties that are traditionally associated with a single networking interface, such as source addresses, DNS configuration, proxy configuration, and gateway addresses.

Clients that are aware of PvDs can take advantage of multiple network interfaces simultaneously. This enables using two PvDs in parallel for separate connections or for multi-path transports.

While most PvDs today are discovered implicitly (such as by receiving information via Router Advertisements from a router on a network that a client host directly connects to), [RFC7556] also defines the notion of Explicit PvDs. IPsec Virtual Private Networks are considered Explicit PvDs, but Explicit PvDs can also be discovered via the local network router. Discovering Explicit PvDs allows two key advancements in managing multiple PvDs:

1. The ability to discover and use multiple PvDs on a single interface, such as when a local router can provide connectivity to two different Internet Service Providers.
2. The ability to associate additional information about PvDs to describe the properties of the network.

While [RFC7556] defines the concept of Explicit PvDs, it does not define the mechanism for discovering multiple Explicit PvDs on a single network and their additional information.

This document specifies a way to identify PvDs with Fully Qualified Domain Names (FQDN), called PvD IDs. Those identifiers are advertised in a new Router Advertisement (RA) [RFC4861] option called the PvD ID Router Advertisement option which, when present, associates the PvD ID with all the information present in the Router Advertisement as well as any configuration object, such as addresses, derived from it. The PVD ID Router Advertisement option may also contain a set of other RA options, along with an optional inner Router Advertisement message header. These options and optional

inner header are only visible to 'PvD-aware' hosts, allowing such hosts to have a specialized view of the network configuration.

Since PvD IDs are used to identify different ways to access the internet, multiple PvDs (with different PvD IDs) can be provisioned on a single host interface. Similarly, the same PvD ID could be used on different interfaces of a host in order to inform that those PvDs ultimately provide equivalent services.

This document also introduces a mechanism for hosts to retrieve optional additional information related to a specific PvD by means of an HTTP over TLS query using a URI derived from the PvD ID. The retrieved JSON object contains additional information that would typically be considered too large to be directly included in the Router Advertisement, but might be considered useful to the applications, or even sometimes users, when choosing which PvD should be used.

For example, if Alice has both a cellular network provider and a broadband provider in her home, her PvD-aware devices and applications would be aware of both available uplinks. These applications could fail-over between these networks, or run connections over both (potentially using multi-path transports). Applications could also select specific uplinks based on the properties of the network; for example, if the cellular network provides free high-quality video streaming, a video-streaming application could select that network while most of the other traffic on Alice's device uses the broadband provider.

### 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 uses the following terminology:

**Provisioning Domain (PvD):** A set of network configuration information; for more information, see [RFC7556].

**PvD ID:** A Fully Qualified Domain Name (FQDN) used to identify a PvD.

**Explicit PvD:** A PvD uniquely identified with a PvD ID. For more information, see [RFC7556].

**Implicit PvD:** A PvD that, in the absence of a PvD ID, is identified by the host interface to which it is attached and the address of the advertising router. See also [RFC7556].

**PvD-aware host:** A host that supports the association of network configuration information into PvDs and the use of these PvDs as described in this document. Also named PvD-aware node in [RFC7556].

### 3. Provisioning Domain Identification using Router Advertisements

Explicit PvDs are identified by a PvD ID. The PvD ID is a Fully Qualified Domain Name (FQDN) that identifies the network operator. Network operators **MUST** use names that they own or manage to avoid naming conflicts. The same PvD ID **MAY** be used in several access networks when they ultimately provide identical services (e.g., in all home networks subscribed to the same service); else, the PvD ID **MUST** be different to follow Section 2.4 of [RFC7556].

#### 3.1. PvD ID Option for Router Advertisements

This document introduces a Router Advertisement (RA) option called the PvD Option. It is used to convey the FQDN identifying a given PvD (see Figure 1), bind the PvD ID with configuration information received over DHCPv4 (see Section 3.4.2), enable the use of HTTP over TLS to retrieve the PvD Additional Information JSON object (see Section 4), as well as contain any other RA options which would otherwise be valid in the RA.

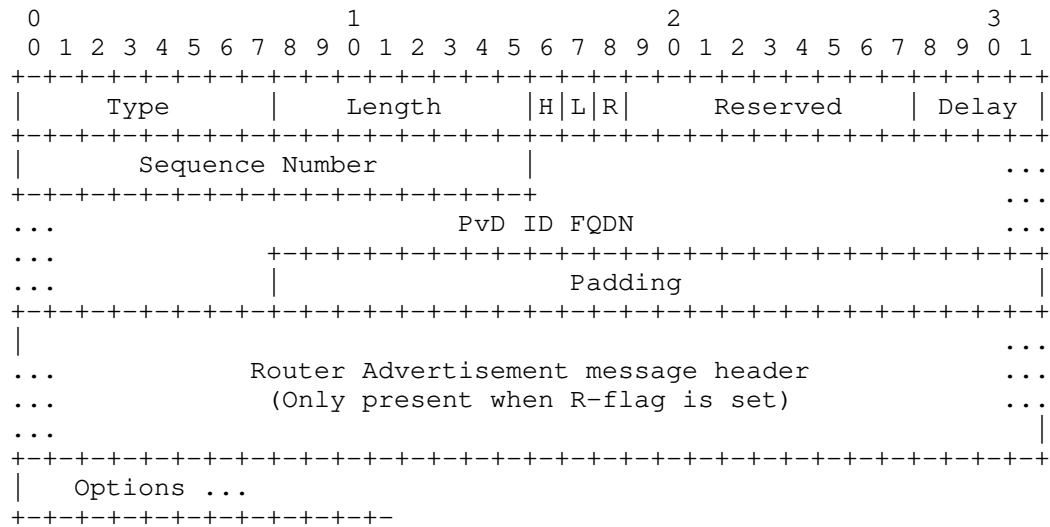


Figure 1: PvD ID Router Advertisements Option Format

Type: (8 bits) Set to 21.

Length: (8 bits) The length of the option in units of 8 octets, including the Type and Length fields, the Router Advertisement message header, if any, as well as the RA options that are included within the PvD Option.

H-flag: (1 bit) 'HTTP' flag stating whether some PvD Additional Information is made available through HTTP over TLS, as described in Section 4.

L-flag: (1 bit) 'Legacy' flag stating whether the PvD is associated with IPv4 information assigned using DHCPv4 (see Section 3.4.2).

R-flag: (1 bit) 'Router Advertisement' flag stating whether the PvD Option header is followed (right after padding to the next 64 bits boundary) by a Router Advertisement message header (see section 4.2 of [RFC4861]). The usage of the inner message header is described in Section 3.4.

Reserved: (13 bits) Reserved for later use. It MUST be set to zero by the sender and ignored by the receiver.

Delay: (4 bits) Unsigned integer used to delay HTTP GET queries from hosts by a randomized backoff (see Section 4.1). If the H-flag is not set, senders SHOULD set the delay to zero, and receivers SHOULD ignore the value.

Sequence Number: (16 bits) Sequence number for the PvD Additional Information, as described in Section 4. If the H-flag is not set, senders SHOULD set the Sequence Number to zero, and receivers SHOULD ignore the value.

PvD ID FQDN: The FQDN used as PvD ID encoded in DNS format, as described in Section 3.1 of [RFC1035]. Domain name compression described in Section 4.1.4 of [RFC1035] MUST NOT be used.

Padding: Zero or more padding octets to the next 8 octet boundary (see Section 4.6 of [RFC4861]). It MUST be set to zero by the sender, and ignored by the receiver.

RA message header: (16 octets) When the R-flag is set, a full Router Advertisement message header as specified in [RFC4861]. The sender MUST set the 'Type' to 134, the value for "Router Advertisement", and set the 'Code' to 0. Receivers MUST ignore both of these fields. The 'Checksum' MUST be set to 0 by the sender; non-zero checksums MUST be ignored by the receiver without causing the processing of the message to fail. All other fields are to be set and parsed as specified in [RFC4861] or any updating documents.

Options: Zero or more RA options that would otherwise be valid as part of the Router Advertisement main body, but are instead included in the PvD Option so as to be ignored by hosts that are not PvD-aware.

Figure 2 shows an example of a PvD Option with "example.org" as the PvD ID FQDN and including both a Recursive DNS Server (RDNSS) option and a prefix information option. It has a Sequence Number of 123, and indicates the presence of additional information that is expected to be fetched with a delay factor of 1.

0										1										2										3																			
0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1	2	3	4	5	6	7	8	9	0	1																		
Type: 21										Length: 12										1 0 0										Reserved										Delay:1									
Seq number: 123										7										e																													
x										a										m										p																			
l										e										3										o																			
r										g										0										0 (padding)																			
0 (padding)										0 (padding)										0 (padding)										0 (padding)																			
RDNSS option (RFC 8106) length: 5																														...																			
...																														...																			
...																																																	
Prefix Information Option (RFC 4861) length: 4																														...																			
...																																																	
...																																																	

Figure 2

### 3.2. Router Behavior

A router MAY send RAs containing one PvD Option, but MUST NOT include more than one PvD Option in each RA. The PvD Option MUST NOT contain further PvD Options.

The PvD Option MAY contain zero, one, or more RA options which would otherwise be valid as part of the same RA. Such options are processed by PvD-aware hosts, while ignored by other hosts as per section 4.2 of [RFC4861].

In order to provide multiple different PvDs, a router MUST send multiple RAs. RAs sent from different link-local source addresses establish distinct implicit PvDs, in the absence of a PvD Option. Explicit PvDs MAY share link-local source addresses with an Implicit PvD and any number of other Explicit PvDs.

In other words, different Explicit PvDs MAY be advertised with RAs using the same link-local source address; but different Implicit PvDs, advertised by different RAs, MUST use different link-local addresses because these Implicit PvDs are identified by the source addresses of the RAs. If a link-local address on the router is

changed, then any new RA will be interpreted as a different Implicit PvD by PvD-aware hosts.

As specified in [RFC4861] and [RFC6980], when the set of options causes the size of an advertisement to exceed the link MTU, multiple router advertisements MUST be sent to avoid fragmentation, each containing a subset of the options. In such cases, the PvD Option header (i.e., all fields except the 'Options' field) MUST be repeated in all the transmitted RAs. The options within the 'Options' field, MAY be transmitted only once, included in one of the transmitted PvD Options.

### 3.3. Non-PvD-aware Host Behavior

As the PvD Option has a new option code, non-PvD-aware hosts will simply ignore the PvD Option and all the options it contains (see section 4.2 of [RFC4861]. This ensures the backward compatibility required in Section 3.3 of [RFC7556]. This behavior allows for a mixed-mode network where a mix of PvD-aware and non-PvD-aware hosts coexist.

### 3.4. PvD-aware Host Behavior

Hosts MUST associate received RAs and included configuration information (e.g., Router Valid Lifetime, Prefix Information [RFC4861], Recursive DNS Server [RFC8106], Routing Information [RFC4191] options) with the Explicit PvD identified by the first PvD Option present in the received RA, if any, or with the Implicit PvD identified by the host interface and the source address of the received RA otherwise. If an RA message header is present both within the PvD Option and outside it, the header within the PvD Option takes precedence.

In case multiple PvD Options are found in a given RA, hosts MUST ignore all but the first PvD Option.

If a host receives PvD Options flags that it does not recognize (currently in the Reserved field), it MUST ignore these flags.

Similarly, hosts MUST associate all network configuration objects (e.g., default routers, addresses, more specific routes, DNS Recursive Resolvers) with the PvD associated with the RA that provisioned the object. For example, addresses that are generated using a received Prefix Information option (PIO) are associated with the PvD of the last received RA which included the given PIO.



PvD IDs MUST be compared in a case-insensitive manner as defined by [RFC4343]. For example, "pvd.example.com." or "PvD.Example.coM." would refer to the same PvD.

While performing PvD-specific operations such as resolving names, executing the default address selection algorithm [RFC6724] or executing the default router selection algorithm when forwarding packets ([RFC4861], [RFC4191] and [RFC8028]), hosts and applications MAY consider only the configuration associated with any non-empty subset of PvDs. For example, a host MAY associate a given process with a specific PvD, or a specific set of PvDs, while associating another process with another PvD. A PvD-aware application might also be able to select, on a per-connection basis, which PvDs should be used. In particular, constrained devices such as small battery operated devices (e.g., IoT), or devices with limited CPU or memory resources may purposefully use a single PvD while ignoring some received RAs containing different PvD IDs.

The way an application expresses its desire to use a given PvD, or a set of PvDs, or the way this selection is enforced, is out of the scope of this document. Useful insights about these considerations can be found in [I-D.kline-mif-mpvd-api-reqs].

#### 3.4.1. DHCPv6 configuration association

When a host retrieves stateless configuration elements using DHCPv6 (e.g., DNS recursive resolvers or DNS domain search lists [RFC3646]), they MUST be associated with all the explicit and implicit PvDs received on the same interface and contained in a RA with the O-flag set [RFC4861].

When a host retrieves stateful assignments using DHCPv6, such assignments MUST be associated with the received PvD which was received with RAs with the M-flag set and including a matching PIO. A PIO is considered to match a DHCPv6 assignment when the IPv6 prefix from the PIO includes the assignment from DHCPv6. For example, if a PvD's associated PIO defines the prefix 2001:db8:cafe::/64, a DHCPv6 IA\_NA message that assigns the address 2001:db8:cafe::1234:4567 would be considered to match.

In cases where an address would be assigned by DHCPv6 and no matching PvD could be found, hosts MAY associate the assigned address with any implicit PvD received on the same interface or to multiple implicit PvDs received on the same interface. This is intended to resolve backward compatibility issues with rare deployments choosing to assign addresses with DHCPv6 while not sending any matching PIO. Implementations are suggested to flag or log such scenarios as errors to help detect misconfigurations.

### 3.4.2. DHCPv4 configuration association

Associating DHCPv4 [RFC2131] configuration elements with Explicit PvDs allows hosts to treat a set of IPv4 and IPv6 configurations as a single PvD with shared properties. For example, consider a router that provides two different uplinks. One could be a broadband network that has data rate and streaming properties described in PvD additional information and that provides both IPv4 and IPv6 network access. The other could be a cellular network that provides only IPv6 network access, and uses NAT64 [RFC6146]. The broadband network can be represented by an Explicit PvD that points to the additional information, and also marks association with DHCPv4 information. The cellular network can be represented by a different Explicit PvD that is not associated with DHCPv4.

When a PvD-aware host retrieves configuration elements from DHCPv4, the information is associated either with a single Explicit PvD on that interface, or else with all Implicit PvDs on the same interface.

An Explicit PvD indicates its association with DHCPv4 information by setting the L-flag in the PvD RA Option. If there is exactly one Explicit PvD that sets this flag, hosts MUST associate the DHCPv4 information with that PvD. Multiple Explicit PvDs on the same interface marking this flag is a misconfiguration, and hosts SHOULD NOT associate the DHCPv4 information with any Explicit PvD in this case.

If no single Explicit PvD claims association with DHCPv4, the configuration elements coming from DHCPv4 MUST be associated with all Implicit PvDs identified by the interface on which the DHCPv4 transaction happened. This maintains existing host behavior.

### 3.4.3. Connection Sharing by the Host

The situation when a host shares connectivity from an upstream interface (e.g., cellular) to a downstream interface (e.g., Wi-Fi) is known as 'tethering'. Techniques such as ND-proxy [RFC4389], 64share [RFC7278] or prefix delegation (e.g., using DHCPv6-PD [RFC8415]) may be used for that purpose.

Whenever the RAs received from the upstream interface contain a PVD RA option, hosts that are sharing connectivity SHOULD include a PVD option within the RAs sent downstream with:

- o The same PVD-ID FQDN
- o The same H-flag, Delay and Sequence Number values

- o The L bit set whenever the host is sharing IPv4 connectivity received from the same upstream interface
- o The bits from the Reserved field set to 0

The values of the R-flag, Router Advertisement message header and Options field depend on whether the connectivity should be shared only with PvD-aware hosts or not (see Section 3.2). In particular, all options received within the upstream PvD Option and included in the downstream RA SHOULD be included in the downstream PvD Option.

#### 3.4.4. Usage of DNS Servers

PvD-aware hosts can be provisioned with recursive DNS servers via RA options passed within an Explicit PvD, via RA options associated with an Implicit PvD, via DHCPv6 or DHCPv4, or from some other provisioning mechanism that creates an Explicit PvD (such as a VPN). In all of these cases, the recursive DNS server addresses SHOULD be associated with the corresponding PvD. Specifically, queries sent to a configured recursive DNS server SHOULD be sent from a local IP address that was provisioned for the PvD via RA or DHCP. Answers received from the DNS server SHOULD only be used on the same PvD.

PvD-aware applications will be able to select which PvD(s) to use for DNS resolution and connections, which allows them to effectively use multiple Explicit PvDs. In order to support non-PvD-aware applications, however, PvD-aware hosts SHOULD ensure that non-PvD-aware name resolution APIs like "getaddrinfo" only use resolvers from a single PvD for a given query. Handling DNS across PvDs is discussed in Section 5.2.1 of [RFC7556], and PvD APIs are discussed in Section 6 of [RFC7556].

Maintaining the correct usage of DNS within PvDs avoids various practical errors, such as:

- o A PvD associated with a VPN or otherwise private network may provide DNS answers that contain addresses inaccessible over another PvD. This includes the DNS queries to retrieve PvD additional information, which could otherwise send identifying information to the recursive DNS system (see Section 4.1).
- o A PvD that uses a NAT64 [RFC6146] and DNS64 [RFC6147] will synthesize IPv6 addresses in DNS answers that are not globally routable, and would be invalid on other PvDs. Conversely, an IPv4 address resolved via DNS on another PvD cannot be directly used on a NAT64 network.

#### 4. Provisioning Domain Additional Information

Additional information about the network characteristics can be retrieved based on the PvD ID. This set of information is called PvD Additional Information, and is encoded as a JSON object [RFC8259]. This JSON object is restricted to the I-JSON profile, as defined in [RFC7493].

The purpose of this JSON object is to provide additional information to applications on a client host about the connectivity that is provided using a given interface and source address. It typically includes data that would be considered too large, or not critical enough, to be provided within an RA option. The information contained in this object MAY be used by the operating system, network libraries, applications, or users, in order to decide which set of PvDs should be used for which connection, as described in Section 3.4.

The additional information related to a PvD is specifically intended to be optional, and is targeted at optimizing or informing the behavior of user-facing hosts. This information can be extended to provide hints for host system behavior (such as captive portal or walled-garden PvD detection) or application behavior (describing application-specific services offered on a given PvD). This content may not be appropriate for light-weight Internet of Things (IoT) devices. IoT devices might need only a subset of the information, and would in some cases prefer a smaller representation like CBOR ([RFC7049]). Delivering a reduced version of the PvD Additional Information designed for such devices is not defined in this document.

##### 4.1. Retrieving the PvD Additional Information

When the H-flag of the PvD Option is set, hosts MAY attempt to retrieve the PvD Additional Information associated with a given PvD by performing an HTTP over TLS [RFC2818] GET query to `https://<PvD-ID>/well-known/pvd`. Inversely, hosts MUST NOT do so whenever the H-flag is not set.

Recommendations for how to use TLS securely can be found in [RFC7525].

When a host retrieves the PvD Additional Information, it MUST verify that the TLS server certificate is valid for the performed request; specifically, that a DNS-ID [RFC6125] on the certificate is equal to the PvD ID expressed as an FQDN. This validation indicates that the owner of the FQDN authorizes its use with the prefix advertised by

the router. If this validation fails, hosts MUST close the connection and treat the PvD as if it has no Additional Information.

HTTP requests and responses for PvD additional information use the "application/pvd+json" media type (see Section 8). Clients SHOULD include this media type as an Accept header field in their GET requests, and servers MUST mark this media type as their Content-Type header field in responses.

Note that the DNS name resolution of the PvD ID, any connections made for certificate validation (such as OCSP [RFC6960]), and the HTTP request itself MUST be performed using the considered PvD. In other words, the name resolution, PKI checks, source address selection, as well as the next-hop router selection MUST be performed while using exclusively the set of configuration information attached with the PvD, as defined in Section 3.4. In some cases, it may therefore be necessary to wait for an address to be available for use (e.g., once the Duplicate Address Detection or DHCPv6 processes are complete) before initiating the HTTP over TLS query. In order to address privacy concerns around linkability of the PvD HTTP connection with future user-initiated connections, if the host has a temporary address per [RFC4941] in this PvD, then it SHOULD use a temporary address to fetch the PvD Additional Information and MAY deprecate the used temporary address and generate a new temporary address afterward.

If the HTTP status of the answer is greater than or equal to 400 the host MUST close its connection and consider that there is no additional PvD information. If the HTTP status of the answer is between 300 and 399, inclusive, it MUST follow the redirection(s). If the HTTP status of the answer is between 200 and 299, inclusive, the response is expected to be a single JSON object.

After retrieval of the PvD Additional Information, hosts MUST remember the last Sequence Number value received in an RA including the same PvD ID. Whenever a new RA for the same PvD is received with a different Sequence Number value, or whenever the expiry date for the additional information is reached, hosts MUST deprecate the additional information and stop using it.

Hosts retrieving a new PvD Additional Information object MUST check for the presence and validity of the mandatory fields specified in Section 4.3. A retrieved object including an expiration time that is already past or missing a mandatory element MUST be ignored.

In order to avoid synchronized queries toward the server hosting the PvD Additional Information when an object expires, object updates are delayed by a randomized backoff time.

- o When a host performs a JSON object update after it detected a change in the PvD Option Sequence Number, it MUST add a delay before sending the query. The target time for the delay is calculated as a random time between zero and  $2^{**}(10 + \text{Delay})$  milliseconds, where 'Delay' corresponds to the 4-bit unsigned integer in the last received PvD Option.
- o When a host last retrieved a JSON object at time A that includes a expiry time B using the "expires" key, and the host is configured to keep the PvD information up to date, it MUST add some randomness into its calculation of the time to fetch the update. The target time for fetching the updated object is calculated as a uniformly random time in the interval  $[(B-A)/2, B]$ .

In the example Figure 2, the delay field value is 1, this means that the host calculates its delay by choosing a uniformly random time between 0 and  $2^{**}(10 + 1)$  milliseconds, i.e., between 0 and 2048 milliseconds.

Since the 'Delay' value is directly within the PvD Option rather than the object itself, an operator may perform a push-based update by incrementing the Sequence Number value while changing the Delay value depending on the criticality of the update and its PvD Additional Information servers capacity.

In addition to adding a random delay when fetching Additional Information, hosts MUST enforce a minimum time between requesting Additional Information for a given PvD on the same network. This minimum time is RECOMMENDED to be 10 seconds, in order to avoid hosts causing a denial-of-service on the PvD server. Hosts also MUST limit the number of requests that are made to different PvD Additional Information servers on the same network within a short period of time. A RECOMMENDED value is to issue no more than five PvD Additional Information requests in total on a given network within 10 seconds. For more discussion, see Section 6.

The PvD Additional Information object includes a set of IPv6 prefixes (under the key "prefixes") which MUST be checked against all the Prefix Information Options advertised in the RA. If any of the prefixes included in any associated PIO is not covered by at least one of the listed prefixes, the associated PvD information MUST be considered to be a misconfiguration, and MUST NOT be used by the host. See Section 4.4 for more discussion on handling such misconfigurations.

If the request for PvD Additional Information fails due to a TLS certificate validation error, an HTTP error, or because the retrieved file does not contain valid PvD JSON, hosts MUST close any connection

used to fetch the PvD Additional Information, and MUST NOT request the information for that PvD ID again for the duration of the local network attachment. If a host detects 10 or more such failures to fetch PvD Additional Information, the local network is assumed to be misconfigured or under attack, and the host MUST NOT make any further requests for any PvD Additional Information, belonging to any PvD ID, for the duration of the local network attachment. For more discussion, see Section 6.

#### 4.2. Operational Consideration to Providing the PvD Additional Information

Whenever the H-flag is set in the PvD Option, a valid PvD Additional Information object MUST be made available to all hosts receiving the RA by the network operator. In particular, when a captive portal is present, hosts MUST still be allowed to perform DNS, certificate validation, and HTTP over TLS operations related to the retrieval of the object, even before logging into the captive portal.

Routers SHOULD increment the PVD Option Sequence Number by one whenever a new PvD Additional Information object is available and should be retrieved by hosts. If the value exceeds what can be stored in the Sequence Number field, it MUST wrap back to zero.

The server providing the JSON files SHOULD also check whether the client address is contained by the prefixes listed in the additional information, and SHOULD return a 403 response code if there is no match.

#### 4.3. PvD Additional Information Format

The PvD Additional Information is a JSON object.

The following table presents the mandatory keys which MUST be included in the object:

JSON key	Description	Type	Example
identifier	PvD ID FQDN	String	"pvd.example.com."
expires	Date after which this object is no longer valid	[RFC3339] Date	"2020-05-23T06:00:00Z"
prefixes	Array of IPv6 prefixes valid for this PvD	Array of strings	["2001:db8:1::/48", "2001:db8:4::/48"]

A retrieved object which does not include all three of these keys at the root of the JSON object MUST be ignored. All three keys need to be validated, otherwise the object MUST be ignored. The value stored for "identifier" MUST be matched against the PvD ID FQDN presented in the PvD RA option using the comparison mechanism described in Section 3.4. The value stored for "expires" MUST be a valid date in the future. If the PIO of the received RA is not covered by at least one of the "prefixes" key, the retrieved object SHOULD be ignored.

The following table presents some optional keys which MAY be included in the object.

JSON key	Description	Type	Example
dnsZones	DNS zones searchable and accessible	Array of strings	["example.com", "sub.example.com"]
noInternet	No Internet, set to "true" when the PvD is restricted.	Boolean	true

It is worth noting that the JSON format allows for extensions. Whenever an unknown key is encountered, it MUST be ignored along with its associated elements.

Private-use or experimental keys MAY be used in the JSON dictionary. In order to avoid such keys colliding with IANA registry keys, implementers or vendors defining private-use or experimental keys



MUST create sub-dictionaries. If a set of PvD Additional Information keys are defined by an organization that has a Formal URN Namespace [URN], the URN namespace SHOULD be used as the top-level JSON key for the sub-dictionary. For other private uses, the sub-dictionary key SHOULD follow the format of "vendor-\*", where the "\*" is replaced by the implementer's or vendor's identifier. For example, keys specific to the FooBar organization could use "vendor-foobar". If a host receives a sub-dictionary with an unknown key, the host MUST ignore the contents of the sub-dictionary.

#### 4.3.1. Example

The following two examples show how the JSON keys defined in this document can be used:

```
{
  "identifier": "cafe.example.com.",
  "expires": "2020-05-23T06:00:00Z",
  "prefixes": ["2001:db8:1::/48", "2001:db8:4::/48"],
}

{
  "identifier": "company.foo.example.com.",
  "expires": "2020-05-23T06:00:00Z",
  "prefixes": ["2001:db8:1::/48", "2001:db8:4::/48"],
  "vendor-foo":
    {
      "private-key": "private-value",
    },
}
```

#### 4.4. Detecting misconfiguration and misuse

Hosts MUST validate the TLS server certificate when retrieving PvD Additional Information, as detailed in Section 4.1.

Hosts MUST verify that all prefixes in all the RA PIOs are covered by a prefix from the PvD Additional Information. An adversarial router attempting to spoof the definition of an Explicit PvD, without the ability to modify the PvD Additional Information, would need to perform NAT66 in order to circumvent this check. Thus, this check cannot prevent all spoofing, but it can detect misconfiguration or mismatched routers that are not adding a NAT.

If NAT66 is being added in order to spoof PvD ownership, the HTTPS server for additional information can detect this misconfiguration. The HTTPS server SHOULD validate the source addresses of incoming connections (see Section 4.1). This check gives reasonable assurance

that neither NPTv6 [RFC6296] nor NAT66 were used and restricts the information to the valid network users. If the PvD does not provision IPv4 (it does not include the 'L' bit in the RA), the server cannot validate the source addresses of connections using IPv4. Thus, the PvD ID FQDN for such PvDs SHOULD NOT have a DNS A record.

## 5. Operational Considerations

This section describes some example use cases of PvDs. For the sake of simplicity, the RA messages will not be described in the usual ASCII art but rather in an indented list. Values in the PvD Option header that are not included in the example are assumed to be zero or false (such as the H-flag, Sequence Number, and Delay fields).

### 5.1. Exposing Extra RA Options to PvD-Aware Hosts

In this example, there is one RA message sent by the router. This message contains some options applicable to all hosts on the network, and also a PvD Option that also contains other options only visible to PvD-aware hosts.

- o RA Header: router lifetime = 6000
- o Prefix Information Option: length = 4, prefix = 2001:db8:cafe::/64
- o PvD Option header: length = 3 + 5 + 4, PvD ID FQDN = example.org., R-flag = 0 (actual length of the header with padding 24 bytes = 3 \* 8 bytes)
  - \* Recursive DNS Server: length = 5, addresses = [2001:db8:cafe::53, 2001:db8:f00d::53]
  - \* Prefix Information Option: length = 4, prefix = 2001:db8:f00d::/64

Note that a PvD-aware host will receive two different prefixes, 2001:db8:cafe::/64 and 2001:db8:f00d::/64, both associated with the same PvD (identified by "example.org."). A non-PvD-aware host will only receive one prefix, 2001:db8:cafe::/64.

### 5.2. Different RAs for PvD-Aware and Non-PvD-Aware Hosts

It is expected that for some years, networks will have a mixed environment of PvD-aware hosts and non-PvD-aware hosts. If there is a need to give specific information to PvD-aware hosts only, then it is RECOMMENDED to send two RA messages, one for each class of hosts. This approach allows for two distinct sets of configuration

information to be sent in a way that will not disrupt non-PvD-aware hosts. It also lowers the risk that a single RA message will approach its MTU limit due to duplicated information.

If two RA messages are sent for this reason, they MUST be sent from two different link-local source addresses (Section 3.2). For example, here is the RA sent for non-PvD-aware hosts:

- o RA Header: router lifetime = 6000 (non-PvD-aware hosts will use this router as a default router)
- o Prefix Information Option: length = 4, prefix = 2001:db8:cafe::/64
- o Recursive DNS Server Option: length = 3, addresses = [2001:db8:cafe::53]
- o PvD Option header: length = 3 + 2, PvD ID FQDN = foo.example.org., R-flag = 1 (actual length of the header 24 bytes = 3 \* 8 bytes)
- \* RA Header: router lifetime = 0 (PvD-aware hosts will not use this router as a default router), implicit length = 2

And here is the RA sent for PvD-aware hosts:

- o RA Header: router lifetime = 0 (non-PvD-aware hosts will not use this router as a default router)
- o PvD Option header: length = 3 + 2 + 4 + 3, PvD ID FQDN = bar.example.org., R-flag = 1 (actual length of the header 24 bytes = 3 \* 8 bytes)
- \* RA Header: router lifetime = 1600 (PvD-aware hosts will use this router as a default router), implicit length = 2
- \* Prefix Information Option: length = 4, prefix = 2001:db8:f00d::/64
- \* Recursive DNS Server Option: length = 3, addresses = [2001:db8:f00d::53]

In the above example, non-PvD-aware hosts will only use the first listed RA sent by their default router and using the 2001:db8:cafe::/64 prefix. PvD-aware hosts will autonomously configure addresses from both PIOs, but will only use the source address in 2001:db8:f00d::/64 to communicate past the first hop router since only the router sending the second RA will be used as default router; similarly, they will use the DNS server 2001:db8:f00d::53 when communicating from this address.

### 5.3. Enabling Multi-homing for PvD-Aware Hosts

In this example, the goal is to have one prefix from one RA be usable by both non-PvD-aware and PvD-aware hosts; and to have another prefix usable only by PvD-aware hosts. This allows PvD-aware hosts to be able to effectively multi-home on the network.

The first RA is usable by all hosts. The only difference for PvD-aware hosts is that they can explicitly identify the PvD ID associated with the RA. PvD-aware hosts will also use this prefix to communicate with non-PvD-aware hosts on the same network.

- o RA Header: router lifetime = 6000 (non-PvD-aware hosts will use this router as a default router)
- o Prefix Information Option: length = 4, prefix = 2001:db8:cafe::/64
- o Recursive DNS Server Option: length = 3, addresses = [2001:db8:cafe::53]
- o PvD Option header: length = 3, PvD ID FQDN = foo.example.org., R-flag = 0 (actual length of the header 24 bytes = 3 \* 8 bytes)

The second RA contains a prefix usable only by PvD-aware hosts. Non-PvD-aware hosts will ignore this RA; hence, the only PvD-aware hosts will be multi-homed.

- o RA Header: router lifetime = 0 (non-PvD-aware hosts will not use this router as a default router)
- o PvD Option header: length = 3 + 2 + 4 + 3, PvD ID FQDN = bar.example.org., R-flag = 1 (actual length of the header 24 bytes = 3 \* 8 bytes)
  - \* RA Header: router lifetime = 1600 (PvD-aware hosts will use this router as a default router), implicit length = 2
  - \* Prefix Information Option: length = 4, prefix = 2001:db8:f00d::/64
  - \* Recursive DNS Server Option: length = 3, addresses = [2001:db8:f00d::53]

Note: the above examples assume that the router has received its PvD IDs from upstream routers or via some other configuration mechanism. Another document could define ways for the router to generate its own PvD IDs to allow the above scenario in the absence of PvD ID provisioning.

#### 5.4. Providing Additional Information to PvD-Aware Hosts

In this example, the router indicates that it provides additional information using the H-flag. The Sequence Number on the PvD Option is set to 7 in this example.

- o RA Header: router lifetime = 6000
- o Prefix Information Option: length = 4, prefix = 2001:db8:cafe::/64
- o Recursive DNS Server Option: length = 3, addresses=[2001:db8:cafe::53]
- o PvD Option header: length = 3, PvD ID FQDN = cafe.example.com., Sequence Number = 7, R-flag = 0, H-flag = 1 (actual length of the header with padding 24 bytes = 3 \* 8 bytes)

A PvD-aware host will fetch `https://cafe.example.com/.well-known/pvd` to get the additional information. The following example shows a GET request that the host sends, in HTTP/2 syntax [RFC7540]:

```
:method = GET
:scheme = https
:authority = cafe.example.com
:path = /.well-known/pvd
accept = application/pvd+json
```

The HTTP server will respond with the JSON additional information:

```
:status = 200
content-type = application/pvd+json
content-length = 116

{
  "identifier": "cafe.example.com.",
  "expires": "2020-05-23T06:00:00Z",
  "prefixes": ["2001:db8:cafe::/48"],
}
```

At this point, the host has the additional information, and knows the expiry time. When either the expiry time passes, or a new Sequence Number is provided in an RA, the host will re-fetch the additional information.

For example, if the router sends a new RA with the Sequence Number set to 8, the host will re-fetch the additional information:

- o PvD Option header: length = 3 + 5 + 4 , PvD ID FQDN =  
cafe.example.com., Sequence Number = 8, R-flag = 0, H-flag = 1  
(actual length of the header with padding 24 bytes = 3 \* 8 bytes)

However, if the router sends a new RA, but the Sequence Number has not changed, the host would not re-fetch the additional information (until and unless the expiry time of the additional information has passed).

## 6. Security Considerations

Since the PvD ID RA option can contain an RA header and other RA options, any security considerations that apply for specific RA options continue to apply when used within a PvD ID option.

Although some solutions such as IPsec or SeND [RFC3971] can be used in order to secure the IPv6 Neighbor Discovery Protocol, in practice actual deployments largely rely on link layer or physical layer security mechanisms (e.g., 802.1x [IEEE8021X]) in conjunction with RA Guard [RFC6105].

If multiple RAs are sent for a single PvD to avoid fragmentation, dropping packets can lead to processing only part of a PvD ID option, which could lead to hosts receiving only part of the contained options. As discussed in Section 3.2, routers MUST include the PvD ID option in all fragments generated.

This specification does not improve the Neighbor Discovery Protocol security model, but simply validates that the owner of the PvD FQDN authorizes its use with the prefix advertised by the router. In combination with implicit trust in the local router (if present), this gives the host some level of assurance that the PvD is authorized for use in this environment. However, when the local router cannot be trusted, no such guarantee is available.

It must be noted that Section 4.4 of this document only provides reasonable assurance against misconfiguration but does not prevent a hostile network access provider from advertising incorrect information that could lead applications or hosts to select a hostile PvD. However, a host that correctly implements the multiple PvD architecture ([RFC7556]) using the mechanism described in this document will be less susceptible to some attacks than a host that does not by being able to check for the various misconfigurations or inconsistencies described in this document.

Since expiration times provided in PvD Additional Information use absolute time, these values can be skewed for hosts without an

accurate time base, or due to clock skew. Such time values MUST NOT be used for security-sensitive functionality or decisions.

An attacker generating RAs on a local network can use the H-flag and the PvD ID to cause hosts on the network to make requests for PvD Additional Information from servers. This can become a denial-of-service attack, in which an attacker can amplify its attack by triggering TLS connections to arbitrary servers in response to sending UDP packets containing RA messages. To mitigate this attack, hosts MUST:

- o limit the rate at which they fetch a particular PvD's Additional Information;
- o limit the rate at which they fetch any PvD Additional Information on a given local network;
- o stop making requests for a PvD ID that does not respond with valid JSON;
- o stop making requests for all PvD IDs once a certain number of failures is reached on a particular network.

Details are provided in Section 4.1. This attack can be targeted at generic web servers, in which case the host behavior of stopping requesting for any server that doesn't behave like a PvD Additional Information server is critical. Limiting requests for a specific PvD ID might not be sufficient if the attacker changes the PvD ID values quickly, so hosts also need to stop requesting if they detect consistent failure when on a network that is under attack. For cases in which an attacker is pointing hosts at a valid PvD Additional Information server (but one that is not actually associated with the local network), the server SHOULD reject any requests that do not originate from the expected IPv6 prefix as described in Section 4.2.

## 7. Privacy Considerations

Retrieval of the PvD Additional Information over HTTPS requires early communications between the connecting host and a server which may be located further than the first hop router. Although this server is likely to be located within the same administrative domain as the default router, this property can't be ensured. To minimize the leakage of identity information while retrieving the PvD Additional Information, hosts SHOULD make use of an IPv6 temporary address and SHOULD NOT include any privacy-sensitive data, such as a User-Agent header field or an HTTP cookie.

Hosts might not always fetch PvD Additional Information, depending on whether or not they expect to use the information. However, if a host whitelisted only certain PvD IDs for which to fetch Additional Information, an attacker could send various PvD IDs in RAs to detect which PvD IDs are whitelisted by the client. To avoid this, hosts SHOULD either fetch Additional Information for all eligible PvD IDs on a given local network, or fetch the information for none of them.

From a user privacy perspective, retrieving the PvD Additional Information is not different from establishing a first connection to a remote server, or even performing a single DNS lookup. For example, most operating systems already perform early queries to static web sites, such as `http://captive.example.com/hotspot-detect.html`, in order to detect the presence of a captive portal.

The DNS queries associated with the PvD Additional Information MUST use the DNS servers indicated by the associated PvD, as described in Section 4.1. This ensures the name of the PvD Additional Information server is not unintentionally sent on another network, thus leaking identifying information about the networks with which the client is associated.

There may be some cases where hosts, for privacy reasons, should refrain from accessing servers that are located outside a certain network boundary. In practice, this could be implemented as a whitelist of 'trusted' FQDNs and/or IP prefixes that the host is allowed to communicate with. In such scenarios, the host SHOULD check that the provided PvD ID, as well as the IP address that it resolves into, are part of the allowed whitelist.

Network operators SHOULD restrict access to PvD Additional Information to only expose it to hosts that are connected to the local network, especially if the Additional Information would provide information about local network configuration to attackers. This can be implemented by whitelisting access from the addresses and prefixes that the router provides for the PvD, which will match the prefixes contained in the PvD Additional Information. This technique is described in Section 4.2.

## 8. IANA Considerations

Upon publication of this document, IANA is asked to remove the 'reclaimable' tag off the value 21 for the PvD Option (from the IPv6 Neighbor Discovery Option Formats registry).



### 8.1. New entry in the Well-Known URIs Registry

IANA is asked to add a new entry in the Well-Known URIs registry [RFC8615] with the following information:

URI suffix: 'pvd'

Change controller: IETF

Specification document: this document

Status: permanent

Related information: N/A

### 8.2. Additional Information PvD Keys Registry

IANA is asked to create and maintain a new registry called "Additional Information PvD Keys", which will reserve JSON keys for use in PvD additional information. The initial contents of this registry are given in Section 4.3, including both the table of mandatory keys and the table of optional keys.

The status of a key as mandatory or optional is intentionally not denoted in the table to allow for flexibility in future use cases. Any new assignments of keys will be considered as optional for the purpose of the mechanism described in this document.

New assignments for Additional Information PvD Keys Registry will be administered by IANA through Expert Review [RFC8126]. Experts are requested to ensure that defined keys do not overlap in names or semantics, and represent non-vendor-specific use cases. Vendor-specific keys SHOULD use sub-dictionaries, as described in Section 4.3.

IANA is asked to place this registry in a new page, entitled "Provisioning Domains (PvDs)".

### 8.3. PvD Option Flags Registry

IANA is also asked to create and maintain a new registry entitled "PvD Option Flags" reserving bit positions from 0 to 12 to be used in the PvD Option bitmask. Bit position 0, 1 and 2 are assigned by this document (as specified in Figure 1). Future assignments require Standards Action [RFC8126].

Since these flags apply to an IPv6 Router Advertisement Option, IANA is asked to place this registry under the existing "Internet Control

Message Protocol version 6 (ICMPv6) Parameters" page, as well as providing a link on the new "Provisioning Domains (PvDs)" page.

#### 8.4. PvD JSON Media Type Registration

This document registers the media type for PvD JSON text, "application/pvd+json".

Type Name: application

Subtype Name: pvd+json

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: Encoding considerations are identical to those specified for the "application/json" media type.

Security considerations: See Section 6.

Interoperability considerations: This document specifies the format of conforming messages and the interpretation thereof.

Published specification: This document

Applications that use this media type: This media type is intended to be used by networks advertising additional Provisioning Domain information, and clients looking up such information.

Fragment identifier considerations: N/A

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: IETF

Change controller: IETF

## 9. Acknowledgments

Many thanks to M. Stenberg and S. Barth for their earlier work: [I-D.stenberg-mif-mpvd-dns], as well as to Basile Bruneau who was author of an early version of this document.

Thanks also to Marcus Keane, Mikael Abrahamsson, Ray Bellis, Zhen Cao, Tim Chown, Lorenzo Colitti, Michael Di Bartolomeo, Ian Farrer, Phillip Hallam-Baker, Bob Hinden, Tatuya Jinmei, Erik Kline, Ted Lemon, Paul Hoffman, Dave Thaler, Suresh Krishnan, Gorrry Fairhurst, Jen Lenkova, Veronika McKillop, Mark Townsley and James Woodyatt for useful and interesting discussions and reviews.

Finally, special thanks to Thierry Danis for his valuable inputs and implementation efforts, Tom Jones for his integration effort into the NEAT project and Rigil Salim for his implementation work.

## 10. References

### 10.1. Normative References

- [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>>.
- [RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <<https://www.rfc-editor.org/info/rfc2818>>.
- [RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <<https://www.rfc-editor.org/info/rfc3339>>.
- [RFC4191] Draves, R. and D. Thaler, "Default Router Preferences and More-Specific Routes", RFC 4191, DOI 10.17487/RFC4191, November 2005, <<https://www.rfc-editor.org/info/rfc4191>>.
- [RFC4343] Eastlake 3rd, D., "Domain Name System (DNS) Case Insensitivity Clarification", RFC 4343, DOI 10.17487/RFC4343, January 2006, <<https://www.rfc-editor.org/info/rfc4343>>.

- [RFC4861] Narten, T., Nordmark, E., Simpson, W., and H. Soliman, "Neighbor Discovery for IP version 6 (IPv6)", RFC 4861, DOI 10.17487/RFC4861, September 2007, <<https://www.rfc-editor.org/info/rfc4861>>.
- [RFC4941] Narten, T., Draves, R., and S. Krishnan, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 4941, DOI 10.17487/RFC4941, September 2007, <<https://www.rfc-editor.org/info/rfc4941>>.
- [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>>.
- [RFC6980] Gont, F., "Security Implications of IPv6 Fragmentation with IPv6 Neighbor Discovery", RFC 6980, DOI 10.17487/RFC6980, August 2013, <<https://www.rfc-editor.org/info/rfc6980>>.
- [RFC7493] Bray, T., Ed., "The I-JSON Message Format", RFC 7493, DOI 10.17487/RFC7493, March 2015, <<https://www.rfc-editor.org/info/rfc7493>>.
- [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <<https://www.rfc-editor.org/info/rfc7525>>.
- [RFC7556] Anipko, D., Ed., "Multiple Provisioning Domain Architecture", RFC 7556, DOI 10.17487/RFC7556, June 2015, <<https://www.rfc-editor.org/info/rfc7556>>.
- [RFC8028] Baker, F. and B. Carpenter, "First-Hop Router Selection by Hosts in a Multi-Prefix Network", RFC 8028, DOI 10.17487/RFC8028, November 2016, <<https://www.rfc-editor.org/info/rfc8028>>.
- [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>>.
- [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>>.

- [RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <<https://www.rfc-editor.org/info/rfc8259>>.
- [RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, <<https://www.rfc-editor.org/info/rfc8615>>.

## 10.2. Informative References

- [I-D.kline-mif-mpvd-api-reqs]  
Kline, E., "Multiple Provisioning Domains API Requirements", draft-kline-mif-mpvd-api-reqs-00 (work in progress), November 2015.
- [I-D.stenberg-mif-mpvd-dns]  
Stenberg, M. and S. Barth, "Multiple Provisioning Domains using Domain Name System", draft-stenberg-mif-mpvd-dns-00 (work in progress), October 2015.
- [IEEE8021X]  
IEEE, "IEEE Standards for Local and Metropolitan Area Networks, Port-based Network Access Control, IEEE Std".
- [RFC2131] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, DOI 10.17487/RFC2131, March 1997, <<https://www.rfc-editor.org/info/rfc2131>>.
- [RFC3646] Droms, R., Ed., "DNS Configuration options for Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3646, DOI 10.17487/RFC3646, December 2003, <<https://www.rfc-editor.org/info/rfc3646>>.
- [RFC3971] Arkko, J., Ed., Kempf, J., Zill, B., and P. Nikander, "Secure Neighbor Discovery (SEND)", RFC 3971, DOI 10.17487/RFC3971, March 2005, <<https://www.rfc-editor.org/info/rfc3971>>.
- [RFC4389] Thaler, D., Talwar, M., and C. Patel, "Neighbor Discovery Proxies (ND Proxy)", RFC 4389, DOI 10.17487/RFC4389, April 2006, <<https://www.rfc-editor.org/info/rfc4389>>.
- [RFC6105] Levy-Abegnoli, E., Van de Velde, G., Popoviciu, C., and J. Mohacsi, "IPv6 Router Advertisement Guard", RFC 6105, DOI 10.17487/RFC6105, February 2011, <<https://www.rfc-editor.org/info/rfc6105>>.

- [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <<https://www.rfc-editor.org/info/rfc6125>>.
- [RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146, April 2011, <<https://www.rfc-editor.org/info/rfc6146>>.
- [RFC6147] Bagnulo, M., Sullivan, A., Matthews, P., and I. van Beijnum, "DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers", RFC 6147, DOI 10.17487/RFC6147, April 2011, <<https://www.rfc-editor.org/info/rfc6147>>.
- [RFC6296] Wasserman, M. and F. Baker, "IPv6-to-IPv6 Network Prefix Translation", RFC 6296, DOI 10.17487/RFC6296, June 2011, <<https://www.rfc-editor.org/info/rfc6296>>.
- [RFC6960] Santesson, S., Myers, M., Ankney, R., Malpani, A., Galperin, S., and C. Adams, "X.509 Internet Public Key Infrastructure Online Certificate Status Protocol - OCSP", RFC 6960, DOI 10.17487/RFC6960, June 2013, <<https://www.rfc-editor.org/info/rfc6960>>.
- [RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <<https://www.rfc-editor.org/info/rfc7049>>.
- [RFC7278] Byrne, C., Drown, D., and A. Vizdal, "Extending an IPv6 /64 Prefix from a Third Generation Partnership Project (3GPP) Mobile Interface to a LAN Link", RFC 7278, DOI 10.17487/RFC7278, June 2014, <<https://www.rfc-editor.org/info/rfc7278>>.
- [RFC7540] Belshe, M., Peon, R., and M. Thomson, Ed., "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <<https://www.rfc-editor.org/info/rfc7540>>.
- [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>>.

- [RFC8415] Mrugalski, T., Siodelski, M., Volz, B., Yourtchenko, A., Richardson, M., Jiang, S., Lemon, T., and T. Winters, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 8415, DOI 10.17487/RFC8415, November 2018, <<https://www.rfc-editor.org/info/rfc8415>>.
- [URN] IANA, "Uniform Resource Names (URN) Namespaces", <<https://www.iana.org/assignments/urn-namespaces/urn-namespaces.xhtml>>.

## Authors' Addresses

Pierre Pfister  
Cisco  
11 Rue Camille Desmoulins  
Issy-les-Moulineaux 92130  
France

Email: [ppfister@cisco.com](mailto:ppfister@cisco.com)

Eric Vyncke  
Cisco  
De Kleetlaan, 6  
Diegem 1831  
Belgium

Email: [evyncke@cisco.com](mailto:evyncke@cisco.com)

Tommy Pauly  
Apple Inc.  
One Apple Park Way  
Cupertino, California 95014  
United States of America

Email: [tpauly@apple.com](mailto:tpauly@apple.com)

David Schinazi  
Google LLC  
1600 Amphitheatre Parkway  
Mountain View, California 94043  
United States of America

Email: [dschinazi.ietf@gmail.com](mailto:dschinazi.ietf@gmail.com)

Wenqin Shao  
Cisco  
11 Rue Camille Desmoulins  
Issy-les-Moulineaux 92130  
France

Email: wenshao@cisco.com



Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: 19 May 2021

S. Jiang  
Huawei Technologies Co., Ltd  
G. Li  
Huawei Technologies  
B. E. Carpenter  
Univ. of Auckland  
15 November 2020

Asymmetric IPv6 for Resource-constrained IoT Networks  
draft-jiang-asymmetric-ipv6-04

Abstract

This document describes a new approach to IPv6 header compression for use in scenarios where minimizing packet size is crucial but routing performance must be maximised.

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 19 May 2021.

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. Proposed Solution . . . . .	3
3. Address Transformation at the Gateway . . . . .	6
4. Routing without Decompression . . . . .	6
5. Address Configuration . . . . .	7
6. Compatibility with Existing Protocols . . . . .	7
7. Relationship to Static Context Header Compression . . . . .	7
8. Security Considerations . . . . .	8
9. IANA Considerations . . . . .	8
10. Acknowledgements . . . . .	8
11. References . . . . .	8
Appendix A. Change log [RFC Editor: Please remove] . . . . .	10
Authors' Addresses . . . . .	10

## 1. Introduction

The large address space of IPv6 is essential for the massive expansion of the network edge that will be caused by "Internet of Things" (IoT) technology over low-power or 5G links. However, the size of a raw IPv6 packet header causes difficulty due to the small maximum transmission units (MTU) allowed by typical low-power, low-cost link layers. For 5G, the importance of header overhead in small packets is discussed in [NGMN-5G]. Thus header compression, including address compression, is an important issue. This decreases the size of raw packets, but compressed IP addresses are not routeable except by decompressing them completely in every forwarding node. There are two issues here. The first is the extra computation resource needed for compressing or decompressing in constrained IoT nodes. The second is that full-length IPv6 routing will consume more memory to store routing tables and packet queues (assuming that routing is not bypassed by tunnelling). Such resource consumption is very undesirable in constrained nodes with limited storage, CPU power, and battery capacity.

To mitigate these issues, here we propose a solution enabling the shortening of IPv6 addresses inside packets, and the routing of packets according to short addresses, without needing the overhead of a decompression step prior to route lookup. Considering that the scale and size of edge networks may vary widely, different lengths of short address can be used in different domains.

As an illustrative example, consider an edge network which is known to never require more than a few hundred nodes, which in most cases will communicate either with each other, or with application layer gateways to the rest of the Internet. Rather than needing 128-bit addresses, such a network could very well operate with 16-bit

addresses. Also, it could very likely operate without needing enhancements such as differentiated services, ECN or flow labels. If only IPv6 is supported, the version number field is pointless. There is no reason for IPv6 packets within such a network to contain 40-byte headers as specified in [RFC8200]. Therefore, the useful information could be carried in 8 bytes (see Figure 1). Furthermore, routers within the edge network can route packets directly on 16-bit addresses, reducing RIB and FIB sizes and the lookup time.

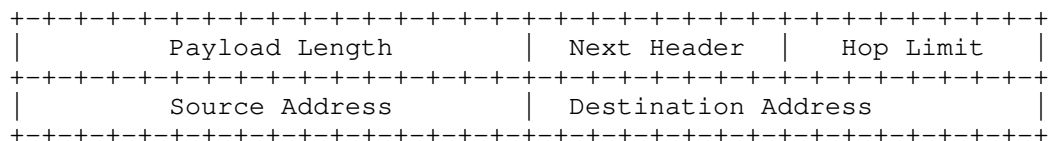


Figure 1

This work is distinct from previous work on address compression [RFC6282] [RFC7400]. Although those solutions tackle the problem of small MTU size, they do not address the problem of decompression overhead.

This work is also distinct from the work on static context header compression [RFC8724], as discussed in more detail below.

Finally, this work is distinct from the 6LoWPAN Routing Header [RFC8138], which can support truncated addresses in a different way.

## 2. Proposed Solution

The use of IPv6 naturally implies 128-bit addresses for both source and destination. However, this address size is huge by the standards of IoT edge networks. We propose the use of a context parameter to indicate the effective length of the IP address for every node in a local domain. If the effective length is N bits, then all addresses in the domain are assumed to be preceded by a common prefix of 128-N bits, when a full size IPv6 address is needed. Any node in the domain that needs the full address, such as a gateway node to the Internet, can therefore easily synthesize it. If a client communicates with a server that is in the local domain, short addresses will be used end-to-end.

The address length parameter may be needed by every node in the domain. It can be spread by various techniques:

- \* Configure the address length in every node.

- \* Obtain the address length from a gateway (next hop router) node.
- \* Negotiate the address length between neighbors.

The solution operates by shortening IP address fields to save overhead. To enhance this, we propose a new field named Flexible Header Encoding (FHE). It consists of 8 bits, each indicating whether the corresponding IPv6 header field [RFC8200] exists.

- \* Bit 0 indicates the Modified Version field
- \* Bit 1 indicates the Traffic Class field
- \* Bit 2 indicates the Flow Label field.
- \* Bit 3 indicates the Payload Length field.
- \* Bit 4 indicates the Next Header field. (Zero implies "No Next Header", value 59)
- \* Bit 5 indicates the Hop Limit field.
- \* Bit 6 indicates the Source Address field.
- \* Bit 7 indicates the Destination Address field.

The "Version" field is a special case. In the context of FHE, all packets are presumed to be IPv6 so the normal version field has no purpose. The Modified Version field, if present, has the following encoded meanings:

- \* 0b0000: The source address (if exist) has pre-determined length inside the domain and the destination address (if exist) uses standard 128-bit IPv6 address. (Outward traffic)
- \* 0b0001: The source address (if exist) uses standard 128-bit IPv6 address and the destination address (if exist) has pre-determined length inside the domain. (Inward traffic)
- \* 0b0010: The source address and destination address have the same length inside the domain. The address length will be pre-determined.
- \* 0b0110: Reserved for IPv6 compatible case.
- \* 0b0100: Reserved for IPv4 compatible case.
- \* 0b0011~0b1111(except 0b0110, 0b0100): Reserved.

All fields, including the Modified Version field, follow the FHE in the same order as in [RFC8200], with no padding. There are no alignment requirements, but when a packet is decompressed to a normal IPv6 format, padding options as defined in RFC8200 must be inserted.

Compared to the illustrative example in Figure 1, the actual packet size would therefore be 10 bytes, a considerable improvement on the standard 40 bytes.

One implication of the above is that the source and destination addresses may be elided completely if they are implicit. Sourceless packets were originally suggested in [Crowcroft].

Figure 2 illustrates an example of the FHE format. In this example the traffic class, flow label and source address are elided, and the destination address is truncated to 16 bits. The modified version field could be 0b0001 or 0b0010.

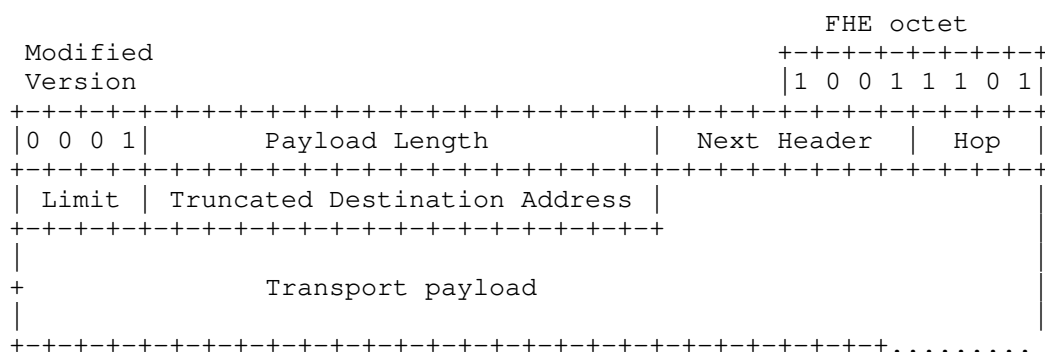


Figure 2

Note that Asymmetric IPv6 does not contain any special handling for IPv6 fragmentation, which will operate exactly as described in [RFC8200], with Asymmetric IPv6 applied to each fragment packet. However, we assume that in IoT deployment scenarios, packets whose length exceeds the IPv6 minimum link MTU before applying Asymmetric IPv6 will be rare. If the underlying link layer cannot carry complete packets even after applying Asymmetric IPv6 compression, an adaptation layer will be necessary exactly as for normal IPv6.

### 3. Address Transformation at the Gateway

Truncated intra-domain addresses will be used to identify nodes inside the domain. When a packet is sent from an IoT node to an external IPv6 host, the node's intra-domain address, which is unique in the domain, will be carried in the source address field. When the packet is forwarded outside the domain by a gateway, the intra-domain address will be transformed to a complete IPv6 address. To achieve this, the gateway should will maintain a globally routeable prefix for all the nodes in the domain. When a packet with an intra-domain source address is received, the gateway extracts this address and concatenates it to the prefix to form a standard, globally unique IPv6 address. Vice versa, when IPv6 packets are received from the Internet, the prefix will be removed to recover the intra-domain short address.

There are two options for handling the addresses of external hosts within the domain. One is to use their full IPv6 addresses via Modified Version codes 0b0000 and 0b0001. The other is effectively a specialized form of Network Address Translation. Here, the gateway will maintain a dynamic mapping table between synthetic intra-domain addresses and IPv6 addresses. As packets are received, the gateway performs the appropriate mapping. The transformation must be checksum-neutral for the transport layer, so the methods designed for NAT46 should be adapted [RFC6145].

It is an engineering choice whether this method is preferable to carrying full 128-bit addresses on the IOT side. Which type of resource is more expensive should be seriously considered to choose the appropriate ways, e.g. computing, memory, or transmitting in various resource-constrained IoT networks.

### 4. Routing without Decompression

Routing mechanisms may readily be adapted to truncated address sizes. If there is routing with an HFE domain, we assume that the truncated address size will be split into a prefix and an interface identifier, but this will not be at the traditional /64 boundary. If routing between different length addresses is required, a suitably modified Forwarding Information Base (FIB) structure is needed, as for any variable length addressing scheme. A truncated address needs to be virtually expanded to 128 bits at the router's inbound interface, although this may not be the physical implementation.

A possible routing choice for IOT edge networks is RPL [RFC6550], although a more complete survey can be found in [Talwar].

## 5. Address Configuration

The simplest approach to address configuration is simply to run normal IPv6 procedures (SLAAC or DHCPv6), on the argument that this is a rare process and the overhead does not matter. If the truncated address size is less than 64 bits, it will be necessary to use shorter interface identifiers than normal, but this is not a major change. Once a node has acquired an IPv6 address and has learned the local address length parameter as outlined in Section 2, it can continue in FHE mode.

## 6. Compatibility with Existing Protocols

Although HFE nodes can only talk directly to each other, they are essentially a special form of IPv6 node and they can communicate with the whole IPv6 Internet via gateways. The complexity is not greater than 6LoWPAN. If appropriate, the 6LoWPAN adaptation layer [RFC4944] could be used, with a specific dispatch type.

## 7. Relationship to Static Context Header Compression

Static Context Header Compression (SCHC) [RFC8724] is a powerful mechanism for reducing IPv6 packet size in an IoT application environment. In particular it includes a profile for UDP over IPv6, and a somewhat modified version of this profile could achieve much of what Asymmetric IPv6 proposes. In addition, SCHC provides support for fragmentation in the case of very small link MTUs. However, SCHC is by design static, and once a context is established the fields to be compressed do not change. Asymmetric IPv6 transmits the FHE and Modified Version bytes with every packet, so it provides dynamic choice as to which header elements are compressed or elided.

In a context where the desirable compression is fixed, e.g. every address is the same length, the flow label is never used, etc., SCHC can be used to the same effect as Asymmetric IPv6. However, if the behavior needs to be dynamic, the signaling power of the FHE and Modified Version bytes in Asymmetric IPv6 is needed.

Further study is needed whether the advantages of the two mechanisms can be combined.

## 8. Security Considerations

HFE is essentially only a non-cryptographic compression technique so it neither adds to nor reduces the intrinsic security of an IPv6 packet. The address length parameter is not a secret, since all nodes in the domain must know it. The mechanism for distributing this parameter must be no less secure than any other configuration mechanism in us.

Address-based privacy issues must be considered in deciding on the address length. If the number of bits available for the interface identifier is significantly less than the 64 currently in use, address traceability and guessability will be affected. However, if the traffic with short addresses is confined to within the edge network, the privacy issue will be minimized. [RFC7721] and [RFC7217] should be consulted prior to deciding the address length.

## 9. IANA Considerations

This document makes no request of the IANA.

NOTE IN DRAFT: If the solution of a 6LoWPAN dispatch type is adopted, a suitable assignment request will be added.

## 10. Acknowledgements

Useful comments were received from Uma Chunduri, Cheng Li, Pascal Thubert, Laurent Toutain and others.

## 11. References

[Crowcroft]

Crowcroft, J. and M. Bagnulo, "SNA: Sourceless Network Architecture", University of Cambridge Computer Laboratory Technical Report UCAM-CL-TR-849, 2014.

[NGMN-5G] Thibault, I., "5G Extreme Requirements: Operators' views on fundamental trade-offs", NGMN Alliance , 2017.

[RFC4944] Montenegro, G., Kushalnagar, N., Hui, J., and D. Culler, "Transmission of IPv6 Packets over IEEE 802.15.4 Networks", RFC 4944, DOI 10.17487/RFC4944, September 2007, <<https://www.rfc-editor.org/info/rfc4944>>.

[RFC6145] Li, X., Bao, C., and F. Baker, "IP/ICMP Translation Algorithm", RFC 6145, DOI 10.17487/RFC6145, April 2011, <<https://www.rfc-editor.org/info/rfc6145>>.



- [RFC6282] Hui, J., Ed. and P. Thubert, "Compression Format for IPv6 Datagrams over IEEE 802.15.4-Based Networks", RFC 6282, DOI 10.17487/RFC6282, September 2011, <<https://www.rfc-editor.org/info/rfc6282>>.
- [RFC6550] Winter, T., Ed., Thubert, P., Ed., Brandt, A., Hui, J., Kelsey, R., Levis, P., Pister, K., Struik, R., Vasseur, JP., and R. Alexander, "RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks", RFC 6550, DOI 10.17487/RFC6550, March 2012, <<https://www.rfc-editor.org/info/rfc6550>>.
- [RFC7217] Gont, F., "A Method for Generating Semantically Opaque Interface Identifiers with IPv6 Stateless Address Autoconfiguration (SLAAC)", RFC 7217, DOI 10.17487/RFC7217, April 2014, <<https://www.rfc-editor.org/info/rfc7217>>.
- [RFC7400] Bormann, C., "6LoWPAN-GHC: Generic Header Compression for IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs)", RFC 7400, DOI 10.17487/RFC7400, November 2014, <<https://www.rfc-editor.org/info/rfc7400>>.
- [RFC7721] Cooper, A., Gont, F., and D. Thaler, "Security and Privacy Considerations for IPv6 Address Generation Mechanisms", RFC 7721, DOI 10.17487/RFC7721, March 2016, <<https://www.rfc-editor.org/info/rfc7721>>.
- [RFC8138] Thubert, P., Ed., Bormann, C., Toutain, L., and R. Cragie, "IPv6 over Low-Power Wireless Personal Area Network (6LoWPAN) Routing Header", RFC 8138, DOI 10.17487/RFC8138, April 2017, <<https://www.rfc-editor.org/info/rfc8138>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.
- [RFC8724] Minaburo, A., Toutain, L., Gomez, C., Barthel, D., and JC. Zúñiga, "SCHC: Generic Framework for Static Context Header Compression and Fragmentation", RFC 8724, DOI 10.17487/RFC8724, April 2020, <<https://www.rfc-editor.org/info/rfc8724>>.
- [Talwar] Talwar, M., "Routing Techniques and Protocols for Internet of Things: a Survey", Indian J.Sci.Res. 12(1):417-423, 2015.

## Appendix A. Change log [RFC Editor: Please remove]

- \* draft-jiang-asymmetric-ipv6-00, 2019-06-03:
  - Initial version
- \* draft-jiang-asymmetric-ipv6-01, 2019-06-21:
  - Fixed reference error
- \* draft-jiang-asymmetric-ipv6-02, 2019-10-29:
  - Added illustrative example
  - Discussed fragmentation
  - Discussed relationship to SCHC
  - Fixed bit pattern errors
- \* draft-jiang-asymmetric-ipv6-03, 2020-05-15:
  - Minor technical and editorial fixes
  - Converted to xml2rfc v3
- \* draft-jiang-asymmetric-ipv6-04, 2020-11-15:
  - Explicitly limit the scope to resource-constrained domain
  - Add engineering choice considerations accordingly

## Authors' Addresses

Sheng Jiang  
Huawei Technologies Co., Ltd  
Q14, Huawei Campus, No.156 Beiqing Road  
Hai-Dian District, Beijing, 100095  
P.R. China

Email: jiangsheng@huawei.com

Guangpeng Li  
Huawei Technologies  
Q14, Huawei Campus  
No.156 Beiqing Road  
Hai-Dian District, Beijing

100095  
P.R. China

Email: [liguangpeng@huawei.com](mailto:liguangpeng@huawei.com)

Brian Carpenter  
The University of Auckland  
School of Computer Science  
University of Auckland  
PB 92019  
Auckland 1142  
New Zealand

Email: [brian.e.carpenter@gmail.com](mailto:brian.e.carpenter@gmail.com)

Network Working Group  
Internet-Draft  
Intended status: Informational  
Expires: 16 November 2020

B. E. Carpenter  
Univ. of Auckland  
S. Jiang  
Huawei Technologies Co., Ltd  
G. Li  
Huawei Technologies  
15 May 2020

Service Oriented Internet Protocol  
draft-jiang-service-oriented-ip-03

Abstract

This document proposes a new, backwards-compatible, approach to packet forwarding, where the service required rather than the IP address required acts as the vector for routing packets at the edge of the network. Deeper in the network, the mechanism can interface to conventional and future methods of service or application aware networking.

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 16 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. Proposed Solution . . . . .	3
3. Coexistence Issues . . . . .	8
4. Some Usage Examples . . . . .	8
5. Continuity with the Existing Internet . . . . .	9
6. Interface with Service and Application Domains . . . . .	9
7. Security Considerations . . . . .	9
8. IANA Considerations . . . . .	10
9. References . . . . .	10
Appendix A. Possible TLV and CBOR Encodings . . . . .	11
A.1. TLV Mapping . . . . .	11
A.2. CBOR Mapping . . . . .	13
Appendix B. Change log [RFC Editor: Please remove] . . . . .	13
Authors' Addresses . . . . .	14

## 1. Introduction

An important aspect of the Internet today is that it is no longer a uniform space with uniform requirements. For both technical and economic reasons, we see an emerging trend of usage scenarios that are confined to some form of limited domain, and which inevitably lead to applications and protocols that are only suitable within a given scope [I-D.carpenter-limited-domains]. In particular, various techniques have emerged for packet treatments that are specific to a type of application or service. This trend collides immediately with two factors: the original design concept of an Internet with end to end IP transparency (such that any locally defined protocol running over IP is almost certain to escape the local network), and with the increasing presence of middleboxes. In this emerging context, where end to end IP service is no longer a safe assumption, and where there is increasing demand for specific services, this document proposes a new, backwards-compatible, approach to packet forwarding, where the service required rather than the IP address required acts as the vector for routing packets at the edge of the network, close to the host requiring a particular service or application. This form of service based packet forwarding is referred to as Service Oriented IP (SOIP).

We propose an addition to the existing core function of the network, which is reachability over IPv6 or IPv4. Today, IP is focussed on reachability, using best effort forwarding both to find a route and to automatically share transmission resources in a simple and low-

cost way. As a result, transport protocols such as TCP and UDP learn little or nothing about the network, beyond congestion or loss signals. Several ISPs may lie on the path between a user and a server, but they are largely ignorant about the services a user requires. Typical services could be streamed content, regular content, user posting, storage access, or calculation, but this list is not exclusive.

Both service providers and users will benefit if a packet stream can be identified intrinsically as requiring a certain kind of service. This is particularly applicable for edge networks, such as those supported by 5G technology, where there is an emphasis on upper layer service provision. Whatever the business model - for example the ISP operates all types of service, or the ISP operates no user services at all and has contracts with specific service providers, or the ISP is agnostic about user services - SOIP will allow for optimised packet delivery. The ISP will have the choice to provide some or all services. The user will have the choice to use ISP services or bypass them. Traffic that leaves the domain where SOIP is in use will be perfectly normal IPv6 or IPv4 traffic, sent by an exit node acting as a proxy (not an IP-layer translator) for the user. Additionally, IPv6 and IPv4 will be modelled as services available to the user, thereby giving continuity of access to everything the user has today. This is a logical extension of a principle already adopted to model IPv4 as a service available via IPv6 [RFC8585].

As new service and application oriented features are deployed in the network, SOIP will provide a seamless interface to both existing and future mechanisms. Effectively it will make client hosts future-proof as the network evolves.

## 2. Proposed Solution

NOTE: This is a preliminary draft expected to stimulate discussion, so many details are not yet defined.

The approach is to make the service be the central component of the network, from the end user's point of view. Conceptually, the user's packets will be directed at a service, not at an IP host. The first hop SOIP router will either forward the packets to an upstream SOIP router, or immediately dispatch the session to a suitable service. At least one SOIP router in a domain must be capable of acting as a dispatcher. The dispatched traffic may either remain in SOIP format, or be transformed by a proxy mechanism into a conventional IP-based format. Figure 1 gives an overview, and Section 5 and Section 6 discuss this further.

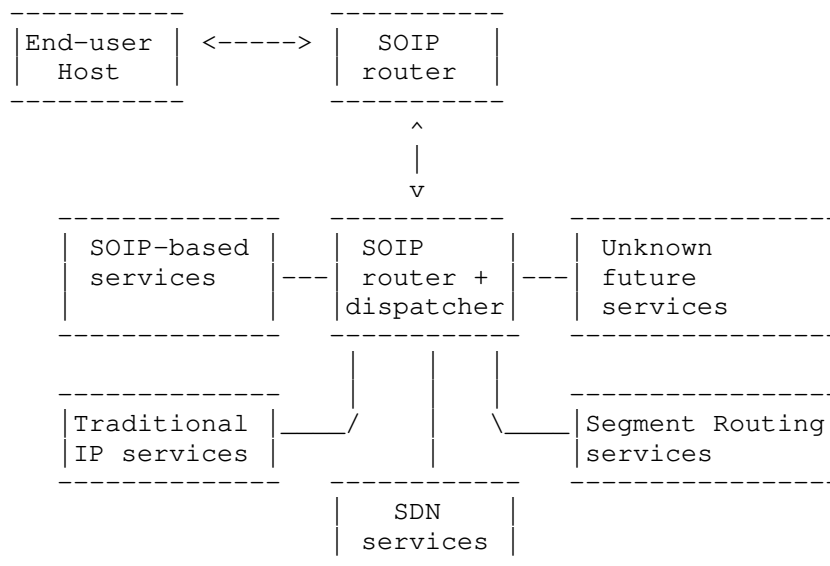


Figure 1

The service actions that the network can provide are abstracted into a number of classes called Service Action Types (SATs). While there needs to be flexibility and extensibility, the number of service action types will be limited. They will not be numerous like IP protocol numbers or well-known TCP or UDP port numbers. Along with the SAT, a source IPv6 address is used to identify the client system. As will be seen below, the destination IPv6 address becomes optional. A consequence is that the IP header and some aspects of the protocol stack have to be redesigned. We will show below how this can be done without disturbing most of the running network. Another consequence is that the first step in processing a packet is to process the SAT, not the destination address (if there is one).

Traditional reachability, when required, is provided by classical IPv6, or by IPv4 as-a-service.

When an SOIP packet enters a router, it is classified at line speed according to the SAT. Routing to upstream SOIP routers will be based on the SAT, not on a destination IP address. Routing from a dispatcher may be based on the SAT if the service required is based on SOIP, or on a conventional IP address otherwise. A preliminary list of SATs is shown in Figure 2, with brief descriptions:

SATs	Direction	Description
IPv4 reachability	Request	IPv4 destination host
	Response	IPv4 source host
IPv6 reachability	Request	IPv6 destination host
	Response	IPv6 source host
Discovery Service	Request	Discover network services, e.g. DNS, CDN. May map to IP Anycast Content ID, service ID. Or "service not available" error
	Response	
Multicast Service	Request	Join multicast service for some content, e.g. video stream
	Response	Multicast directory answers request, provides m/c source.
Computation Service	Request	Submit task to network, with computation type, task descriptor and requester ID Computation resource ID, or direct result of task. Or "service not available" error
	Response	
Storage Service	Request	Submit/retrieve data to/from network storage, with data description and/or data ID Storage locator, data ID. Or "service not available" error
	Response	
App Server Service	Request	To submit source code or deploy package to application platform, with necessary configurations. Answer with service ID. Or "service not available" error
	Response	

Figure 2



For each request there will be a corresponding response. The details remain to be worked out – probably a generic response message including the SAT. To allow multiple overlapping sessions, each request/response sequence should have a unique ID, which will be used by the SOIP dispatcher to match service responses to the appropriate user session.

For IPv6-only networks, is expected that IPv4 reachability will be provided by a solution such as 464XLAT. Also, no separate SAT is needed for IPv6 to IPv4 translation. For example, if a host requests IPv4 reachability but supplies an IPv6 address as its own locator, NAT64 [RFC6146] is implied.

For the moment codes for the SATs are undefined, but they are assumed to be small integers. There are two possible approaches to the packet format. One is to use a traditional Type-Length-Value (TLV) layout. Another is to use a more flexible encoding at the lowest level, taking advantage of some form of network processor in the routers. An obvious choice would be Concise Binary Object Representation (CBOR) [RFC7049], which combines flexibility with efficiency. In either case we could require that the first four bits of the wire format are a new IP version number other than 4 or 6. An alternative, at least for early experimentation, is to run SOIP over UDP and IPv6.

Examples of both encoding choices are described below. In either case, the essential content of a packet header is as follows:

- \* The SAT code (small integer)
- \* Flag bits
- \* Traffic class (as for IPv6)
- \* Session Identifier (so that sessions can be tracked regardless of IP address)
- \* Hop limit (small integer)
- \* User locator (IP address or identifier)
- \* Service data length (not needed in CBOR version)
- \* Service data (length depends on SAT)
- \* Payload length (not needed in CBOR version)
- \* Payload

Experience with IPv4 options, and IPv6 extension headers and options, has shown that new ones are very hard to deploy on an operational network, and that the ones defined during the initial design are not always useful. Therefore we propose that all options and extensions are defined as part of the service data and are not visible as part of the basic packet header, giving good flexibility.

We propose to include the exact equivalent of the IPv6 Traffic Class [RFC8200], which can work exactly as for IPv6. In contrast, one defect in the IPv6 flow label [RFC6437] is that it is different in the two directions of a flow. Instead we propose a session ID that is the same in both directions, which has various advantages by allowing immediate session identification.

The flag bits provide useful indications to the routing system, if set:

- \* Mobile - set if the user system is mobile
- \* Flow size (3 bits)
  - 000 means a single packet, no flow/congestion state needed
  - other values TBD indicate type of flow/congestion state
- \* Authenticated - set if packet authenticated (details TBD)
- \* Encrypted - set if encryption applied (details TBD)

Note that fragmentation is not supported. Fragmentation, and the related mechanisms of MTU discovery, are a significant operational problem in the current Internet [I-D.ietf-intarea-frag-fragile]. We simply abolish this problem area in SOIP, which is designed for use in managed networks where a single size of maximum transmission unit (MTU) is available everywhere. An SOIP network will have a globally defined MTU. Of course, IPv4 and IPv6 reachability services via the open Internet will have to support PMTUD and fragmentation as best they can, but this concerns the embedded IP packets, not the SOIP packets, and will be invisible locally.

Appendix A outlines possible TLV and CBOR encodings of the SOIP protocol.

### 3. Coexistence Issues

SOIP is expected to coexist with IPv6; in a sense it is a low-level method of orchestrating IPv6 connections. We assume that each SOIP client host has at least one IPv6 address, and that SOIP routers will announce themselves using a suitable IPv6 Router Advertisement extension [I-D.troan-6man-universal-ra-option]. Normally, the first-hop SOIP router will be the same as the IPv6 first-hop router.

As a result of this, all standard management mechanisms such as NETCONF may be used without further specification. Also, when a data connection of any kind is established after a SOIP request/response exchange, all standard transport mechanisms are available over IPv6. As noted above, they are subject to the locally defined MTU as long as they remain within the SOIP domain.

We do not define how SOIP would operate in an IPv4-only network.

### 4. Some Usage Examples

\* Storage request (upload content):

- Service data identifies storage requirement (temporary/permanent, private/public, encryption, etc.)
- Payload identifies data (path/name.format, etc.)

\* Storage request (download content):

- Service data identifies transmission requirement (streamed, block, etc. and the specific transport protocol - UDP, TCP, QUIC, etc. - if needed)
- Payload identifies data (path/name.format, etc.)

\* Computation request

- Service data identifies computing requirement
- Payload identifies computing application

\* Reachability request

- Service data gives destination IP address (or DNS name)
- Indication of transport protocol required (details TBD)

- Indication of options or extension headers required (details TBD)

## 5. Continuity with the Existing Internet

Continuity is provided in two ways:

1. A user node can simply use IP completely in parallel with SOIP. The network stack in the user node will simply encode the IP packets as SOIP packets with the SAT for IP reachability, and a SOIP dispatcher will send and receive IP packets at the SOIP domain boundary.
2. If a service in the SOIP domain needs service from elsewhere in the IP Internet to respond to a user request, it will use a similar dispatcher function to do so. This could also be described as a proxy mechanism. (Of course, services in interconnected SOIP domains may talk to each other directly.)

## 6. Interface with Service and Application Domains

Various techniques are emerging for service or application specific networking within operators' networks. An overview of the motivations is given in [I-D.li-apn6-problem-statement-usecases], and specific techniques have been defined such as Network Service Headers [RFC8300] and Segment Routing [RFC8402], as well as Software-Defined Networking in general [RFC7426]. A SOIP dispatcher that is aware of such techniques may convert SOIP traffic into one of these mechanisms, for example by encapsulation or proxying. Furthermore, the model is future-proof. The dispatcher could be upgraded to support unknown future service or application oriented networking mechanisms, without requiring changes to SOIP clients or routers.

## 7. Security Considerations

It is intended that both authentication and encryption should be available for all SOIP packets. However, this requires further work, especially to determine whether existing mechanisms for key management can be used.

Since clients are identified by an IPv6 address, existing layer 3 privacy considerations for IPv6 addresses will apply to SOIP [RFC7721]. Upper layer privacy considerations will depend on the service concerned.

## 8. IANA Considerations

This document makes no request of the IANA.

## 9. References

[I-D.carpenter-limited-domains]

Carpenter, B. and B. Liu, "Limited Domains and Internet Protocols", Work in Progress, Internet-Draft, draft-carpenter-limited-domains-13, 2 February 2020, <<https://tools.ietf.org/html/draft-carpenter-limited-domains-13>>.

[I-D.ietf-intarea-frag-fragile]

Bonica, R., Baker, F., Huston, G., Hinden, R., Troan, O., and F. Gont, "IP Fragmentation Considered Fragile", Work in Progress, Internet-Draft, draft-ietf-intarea-frag-fragile-17, 30 September 2019, <<https://tools.ietf.org/html/draft-ietf-intarea-frag-fragile-17>>.

[I-D.li-apn6-problem-statement-usecases]

Li, Z., Peng, S., Voyer, D., Xie, C., Liu, P., Liu, C., Ebisawa, K., Previdi, S., and J. Guichard, "Problem Statement and Use Cases of Application-aware IPv6 Networking (APN6)", Work in Progress, Internet-Draft, draft-li-apn6-problem-statement-usecases-01, 3 November 2019, <<https://tools.ietf.org/html/draft-li-apn6-problem-statement-usecases-01>>.

[I-D.troan-6man-universal-ra-option]

Troan, O., "The Universal IPv6 Configuration Option (experiment)", Work in Progress, Internet-Draft, draft-troan-6man-universal-ra-option-02, 2 April 2020, <<https://tools.ietf.org/html/draft-troan-6man-universal-ra-option-02>>.

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, DOI 10.17487/RFC6146, April 2011, <<https://www.rfc-editor.org/info/rfc6146>>.

[RFC6437] Amante, S., Carpenter, B., Jiang, S., and J. Rajahalme, "IPv6 Flow Label Specification", RFC 6437, DOI 10.17487/RFC6437, November 2011, <<https://www.rfc-editor.org/info/rfc6437>>.

- [RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <<https://www.rfc-editor.org/info/rfc7049>>.
- [RFC7426] Haleplidis, E., Ed., Pentikousis, K., Ed., Denazis, S., Hadi Salim, J., Meyer, D., and O. Koufopavlou, "Software-Defined Networking (SDN): Layers and Architecture Terminology", RFC 7426, DOI 10.17487/RFC7426, January 2015, <<https://www.rfc-editor.org/info/rfc7426>>.
- [RFC7721] Cooper, A., Gont, F., and D. Thaler, "Security and Privacy Considerations for IPv6 Address Generation Mechanisms", RFC 7721, DOI 10.17487/RFC7721, March 2016, <<https://www.rfc-editor.org/info/rfc7721>>.
- [RFC8200] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", STD 86, RFC 8200, DOI 10.17487/RFC8200, July 2017, <<https://www.rfc-editor.org/info/rfc8200>>.
- [RFC8300] Quinn, P., Ed., Elzur, U., Ed., and C. Pignataro, Ed., "Network Service Header (NSH)", RFC 8300, DOI 10.17487/RFC8300, January 2018, <<https://www.rfc-editor.org/info/rfc8300>>.
- [RFC8402] Filss, C., Ed., Previdi, S., Ed., Ginsberg, L., Decraene, B., Litkowski, S., and R. Shakir, "Segment Routing Architecture", RFC 8402, DOI 10.17487/RFC8402, July 2018, <<https://www.rfc-editor.org/info/rfc8402>>.
- [RFC8585] Palet Martinez, J., Liu, H. M.-H., and M. Kawashima, "Requirements for IPv6 Customer Edge Routers to Support IPv4-as-a-Service", RFC 8585, DOI 10.17487/RFC8585, May 2019, <<https://www.rfc-editor.org/info/rfc8585>>.
- [RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <<https://www.rfc-editor.org/info/rfc8610>>.

## Appendix A. Possible TLV and CBOR Encodings

### A.1. TLV Mapping

Figure 3 shows a possible type-length-value packet format.

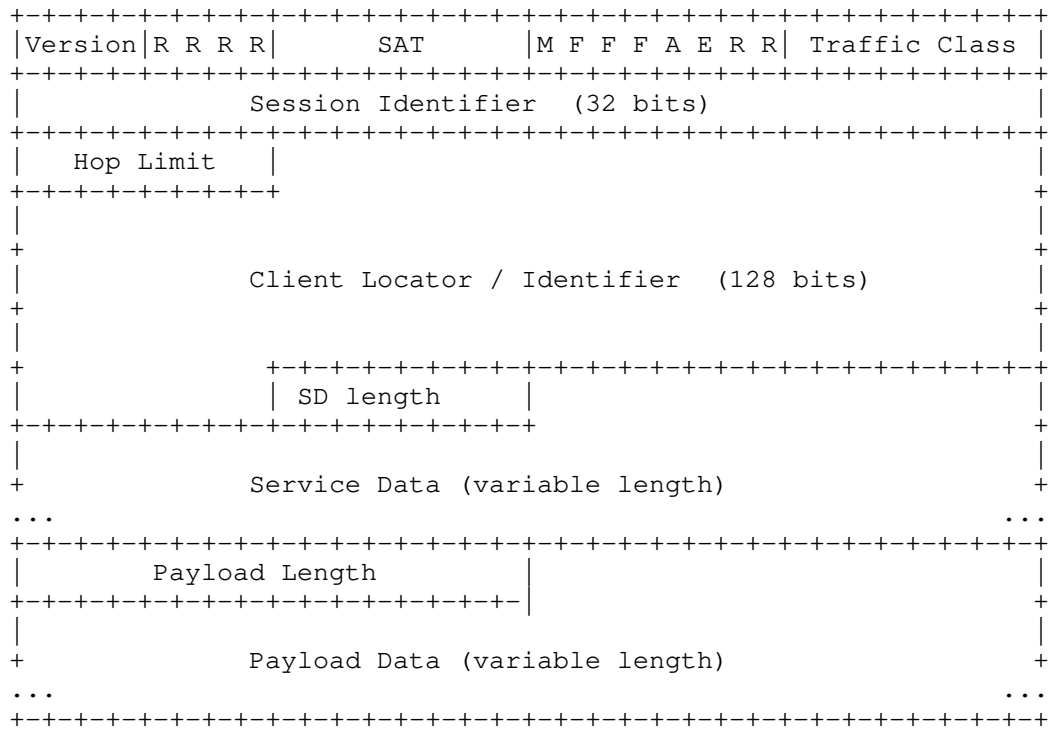


Figure 3

Version - IP version number TBD (not needed over UDP)

R - Reserved, must be zero (not needed over UDP)

SAT - Service Action Type code

M - Mobile flag

FFF - Flow type flags (FFF = 000 for single-packet flows; other values for longer flows)

A - Authentication flag

E - Privacy (encryption) flag

Traffic class, exactly as for IPv6

Session identifier, 32-bit pseudo random number

Client Locator / Identifier - globally unique IPv6 address

## A.2. CBOR Mapping

The packet consists of a CBOR byte string preceded by a single byte (Figure 4). For example, for version 7, this byte would be 0x70. This byte is not decoded as CBOR, and is not needed over UDP.

```

+---+---+---+---+---+---+
|Version|R R R R|
+---+---+---+---+---+

```

Figure 4

The CBOR bytes then obey the CDDL [RFC8610] specification in Figure 5.

```

sat-packet = [sat, flags, traffic-class, session-id, hop-limit,
              source, service-data, ?payload]

```

```

sat = 0..255
flags = bytes .size 1
traffic-class = 0..255
session-id = 0..4294967295 ;up to 32 bits
hop-limit = 0..255
client = ipv6-address
service-data = any
payload = any

```

```

ipv6-address = bytes .size 16

```

Figure 5

The syntax of the various service-data formats can be defined in separate documents for each SAT value.

We assume that routers capable of handling a CBOR-based layer 3 protocol will exist, and will use some form of programmable network processor rather than traditional ASIC or FPGA designs. This allows great flexibility and software-friendly extensibility, especially of the service data formats. Further investigation is needed whether this is realistic.

## Appendix B. Change log [RFC Editor: Please remove]

- \* draft-jiang-service-oriented-ip-00, 2019-05-07:
  - Initial version
- \* draft-jiang-service-oriented-ip-01, 2019-06-21:



- Editorial corrections
- \* draft-jiang-service-oriented-ip-02, 2019-10-29:
  - Added overview diagram
  - Added discussion of dispatcher function
  - Clarifications and editorial corrections
- \* draft-jiang-service-oriented-ip-03, 2020-05-15:
  - Editorial corrections
  - Converted to xml2rfc v3

#### Authors' Addresses

Brian Carpenter  
The University of Auckland  
School of Computer Science  
University of Auckland  
PB 92019  
Auckland 1142  
New Zealand  
  
Email: brian.e.carpenter@gmail.com

Sheng Jiang  
Huawei Technologies Co., Ltd  
Q14, Huawei Campus, No.156 Beiqing Road  
Hai-Dian District, Beijing, 100095  
P.R. China  
  
Email: jiangsheng@huawei.com

Guangpeng Li  
Huawei Technologies  
Q14, Huawei Campus  
No.156 Beiqing Road  
Hai-Dian District, Beijing  
100095  
P.R. China  
  
Email: liguangpeng@huawei.com

TSVWG  
Internet-Draft  
Intended status: Informational  
Expires: January 8, 2020

Y. Li  
X. Zhou  
Huawei  
M. Boucadair  
Orange  
J. Wang  
China Telecom  
July 07, 2019

LOOPS (Localized Optimizations on Path Segments) Problem Statement and  
Opportunities for Network-Assisted Performance Enhancement  
draft-li-tsvwg-loops-problem-opportunities-03

Abstract

In various network deployments, end to end forwarding paths are partitioned into multiple segments. For example, in some cloud-based WAN communications, stitching multiple overlay tunnels are used for traffic policy enforcement matters such as to optimize traffic distribution or to select paths exposing a lower latency. Likewise, in satellite communications, the communication path is decomposed into two terrestrial segments and a satellite segment. Such long-haul paths are naturally composed of multiple network segments with various encapsulation schemes. Packet loss may show different characteristics on different segments.

Traditional transport protocols (e.g., TCP) respond to packet loss slowly especially in long-haul networks: they either wait for some signal from the receiver to indicate a loss and then retransmit from the sender or rely on sender's timeout which is often quite long. Non-congestive loss may make the TCP sender over-reduce the sending rate unnecessarily. With the increase of end-to-end transport encryption (e.g., QUIC), traditional PEP (performance enhancing proxy) techniques such as TCP splitting are no longer applicable.

LOOPS (Local Optimizations on Path Segments) is a network-assisted performance enhancement over path segment and it aims to provide local in-network recovery to achieve better data delivery by making packet loss recovery faster and by avoiding the senders over-reducing their sending rate. In an overlay network scenario, LOOPS can be performed over a variety of the existing, or purposely created, tunnel-based path segments.

## 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 January 8, 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. The Problem . . . . .	3
1.2. Sketching a Work Direction: Rationale & Goals . . . . .	4
2. Terminology . . . . .	6
3. Cloud-Internet Overlay Network . . . . .	7
3.1. Tail Loss or Loss in Short Flows . . . . .	9
3.2. Packet Loss in Real Time Media Streams . . . . .	9
3.3. Packet Loss and Congestion Control in Bulk Data Transfer . . . . .	10
3.4. Multipathing . . . . .	10
4. Satellite Communication . . . . .	11
5. Branch Office WAN Connection . . . . .	13
6. Features and Impacts to be Considered for LOOPS . . . . .	14
6.1. Local Recovery and End-to-end Retransmission . . . . .	15
6.1.1. OE to OE Measurement, Recovery, and Multipathing . . . . .	17

6.2. Congestion Control Interaction . . . . .	18
6.3. Overlay Protocol Extensions . . . . .	19
6.4. Summary . . . . .	20
7. Security Considerations . . . . .	20
8. IANA Considerations . . . . .	21
9. Acknowledgements . . . . .	21
10. Informative References . . . . .	21
Authors' Addresses . . . . .	24

## 1. Introduction

### 1.1. The Problem

Tunnels are widely deployed within many networks to achieve various engineering goals, including long-haul WAN interconnection or enterprise wireless access networks. A connection between two endpoints can be decomposed into many connection legs. As such, the corresponding forwarding path can be partitioned into multiple path segments that some of them are using network overlays by means of tunnels. This design serves a number of purposes such as steering the traffic, optimize egress/ingress link utilization, optimize traffic performance metrics (such as delay, delay variation, or loss), optimize resource utilization by invoking resource bonding, provide high-availability, etc.

A reliable transport layer normally employs some end-to-end retransmission mechanisms which also address congestion control [RFC0793] [RFC5681]. The sender either waits for the receiver to send some signals on a packet loss or sets some form of timeout for retransmission. For unreliable transport protocols such as RTP [RFC3550], optional and limited usage of end-to-end retransmission is employed to recover from packet loss [RFC4585] [RFC4588].

End-to-end retransmission to recover lost packets is slow especially when the network is long-haul. When a path is partitioned into multiple path segments that are realized typically as overlay tunnels, LOOPS (Local Optimizations on Path Segments) aims to provide local segment based in-network recovery to achieve better data delivery by making packet loss recovery faster and by avoiding the senders over-reducing their sending rate. In an overlay network scenario, LOOPS can be performed over the existing, or purposely created, overlay tunnel based path segments. Figure 1 show a basic usage scenario of LOOPS.

Some link types (satellite, microwave, drone-based networking, etc.) may exhibit unusually high loss rate in special conditions (e.g., fades due to heavy rain). The traditional TCP sender interprets loss as congestion and over-reduces the sending rate, degrading the

throughput. LOOPS is also applicable to such scenarios to improve the throughput.

Also, multiple paths may be available in the network that may be used for better performance. These paths are not visible to endpoints. Means to make use of these paths while ensuring the overall performance is enhanced would contribute to customer satisfaction. Blindly implementing link aggregation may lead to undesired effects (e.g., underperform compared to single path).

## 1.2. Sketching a Work Direction: Rationale & Goals

This document sketches a proposal that is meant to experimentally investigate to what extent a network-assisted approach can contribute to increase the overall perceived quality of experience in specific situations (e.g., Sections 3.5 and 3.6 of [RFC8517]) without requiring access to internal transport primitives. The rationale beneath this approach is that some information (loss detection, better visibility on available paths and their characteristics, etc.) can be used to trigger local actions while avoiding as much as possible undesired side effects (e.g., expose a behavior that would be interpreted by an endpoint as an anomaly (corrupt data) and which would lead to exacerbate end-to-end recovery. Such local actions would have a faster effect (e.g., faster recovery, used multiple paths simultaneously).

To that aim, the work is structured into two (2) phased stages:

- o Stage 1: Network-assisted optimization. This one assumes that optimizations (e.g., support latency-sensitive applications) can be implemented at the network without requiring defining new interaction with the endpoint. Existing tools such as ECN will be used. Some of these optimizations may be valuable in deployments where communications are established over paths that are not exposing the same performance characteristics.
- o Stage 2: Collaborative networking optimization. This one requires more interaction between the network and an endpoint to implement coordinated and more surgical network-assisted optimizations based on information/instructions shared by an endpoint or sharing locally-visible information with endpoint for better and faster recovery.

The document focuses on the first stage. Effort related to the second stage is out of scope of the initial planned work. Nevertheless, future work will be planned once progress is (hopefully) made on the first stage.

The proposed mechanism is not meant to be applied to all traffic, but only to a subset which is eligible to the network-assisted optimization service.

Which traffic is eligible is deployment-specific and policy-based. For example, techniques for dynamic information of optimization function (e.g., SFC) may be leveraged to unambiguously identify the aggregate of traffic that is eligible to the service. Such identification may be triggered by subscription actions made by customers or be provided by a network provider (e.g., specific-applications, during specific events such as during severe DDoS attack or flash crowds events).

Likewise, whether the optimization function is permanently instantiated or on-demand is deployment-specific.

This document does not intend to provide a comprehensive list of target deployment cases. Sample scenarios are described to illustrate some LOOPS potentials. Similar issues and optimizations may be helpful in other deployments such as enhancing the reliability of data transfer when a fleet of drones are used for specific missions (e.g., site inspection, live streaming, and emergency service). Captured data should be reliably transmitted via paths involving radio connections.

It is not required that all segments are LOOPS-aware to benefit from LOOPS advantages.

Section 3 presents some of the issues and opportunities found in Cloud-Internet overlay networks that require higher performance and more reliable packet transmission over best effort networks. Section 4 discusses applications of LOOPS in satellite communication. Section 6 describes the corresponding solution features and their impact on existing network technologies.

ON=overlay node  
UN=underlay node

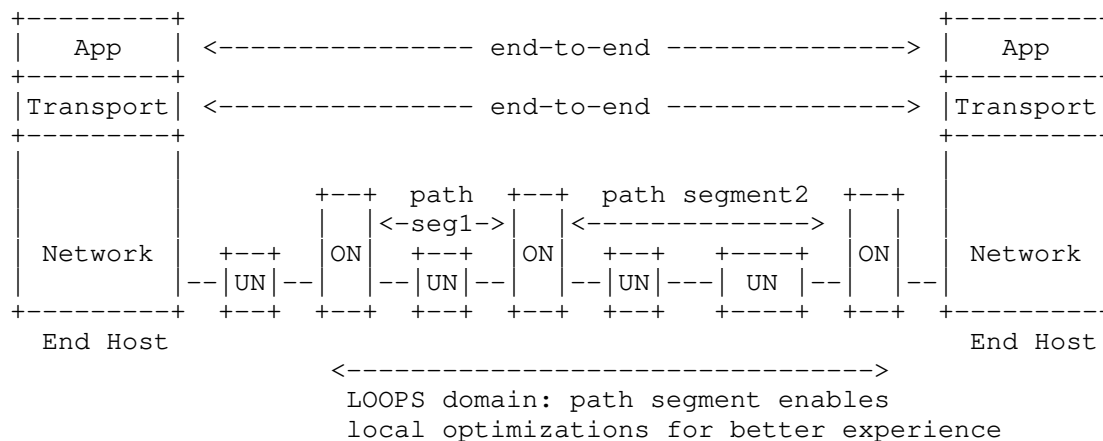


Figure 1: LOOPS in Overlay Network Usage Scenario

## 2. Terminology

This document makes use of the following terms:

LOOPS: Local Optimizations on Path Segments. LOOPS includes to the local in-network (i.e., non end-to-end) recovery functions and other supporting features such as local measurement, loss detection, and congestion feedback.

LOOPS Node: A node supporting LOOPS functions.

Overlay Node (ON): A node having overlay functions (e.g., overlay protocol encapsulation/decapsulation, header modification, TLV inspection) and LOOPS functions in LOOPS overlay network usage scenario.

Overlay Tunnel: A tunnel with designated ingress and egress nodes using some network overlay protocol as encapsulation, optionally with a specific traffic type.

Overlay Edge (OE): Edge node of an overlay tunnel. It can behave as ingress or egress as a function of the traffic direction.

Path segment: A LOOPS enabled tunnel-based network subpath. It is used interchangeably with overlay segment in this document when the context wants to emphasize on its overlay encapsulated nature. It is also called segment for simplicity in this document.

Overlay segment: Refers to path segment.

Underlay Node (UN): A node not participating in the overlay network.

### 3. Cloud-Internet Overlay Network

CSPs (Cloud Service Providers) are connecting their data centers using the Internet or via self-constructed networks/links. This expands the traditional Internet's infrastructure and, together with the original ISP's infrastructure, forms the Internet underlay.

Automation techniques and NFV (Network Function Virtualization) further ambitions to make it easier to dynamically provision a new virtual node/function as a workload in a cloud for CPU/storage intensive functions. With the aid of various mechanisms such as kernel bypassing and Virtual IO, forwarding based on virtual nodes is becoming more and more effective. The interconnection among the purposely positioned virtual nodes and/or the existing nodes with virtualization functions potentially form an overlay infrastructure. It is called the Cloud-Internet Overlay Network (CION) in this document for short.

This architecture scenario makes use of overlay technologies to direct the traffic going through the specific overlay path regardless of the underlying physical topology, in order to achieve better service delivery. It purposely creates or selects overlay nodes (ON) from providers. By continuously measuring the delay of path segments and use them as metrics for path selection, when the number of overlay nodes is sufficiently large, there is a high chance that a better path could be found [DOI\_10.1109\_ICDCS.2016.49] [DOI\_10.1145\_3038912.3052560]. [DOI\_10.1145\_3038912.3052560] further shows all cloud providers experience random loss episodes and random loss accounts for more than 35% of total loss.

Some of the considerations that are discussed below may also apply for interconnecting DCs owned by a network provider.

Figure 2 shows an example of an overlay path over large geographic distances. Three path segments, i.e., ON1-ON2, ON2-ON3, ON3-ON4 are shown. ON is usually a virtual node, though it does not have to be. Each segment transmits packets using some form of network overlay protocol encapsulation. ON has the computing and memory resources that can be used for some functions like packet loss detection, network measurement and feedback, and packet recovery. ONs are managed by a single administrator though they can be workloads created from different CSPs.



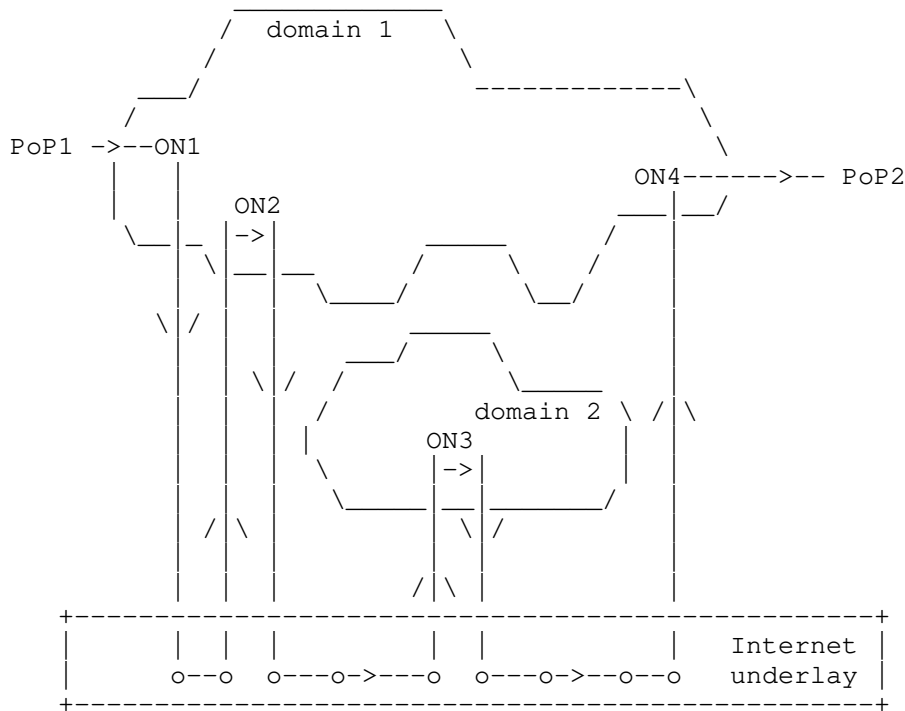


Figure 2: Cloud-Internet Overlay Network (CION)

We tested based on 37 overlay nodes from multiple cloud providers globally. Each pair of the overlay nodes are used as sender and receiver. When the traffic is not intentionally directed to go through any intermediate virtual nodes, we call the path followed by the traffic in the test as the default path. When any of the virtual nodes is intentionally used as an intermediate node to forward the traffic, the path that the traffic takes is called an overlay path. The preliminary experiments showed that the delay of an overlay path is shorter than the one of the default path in 69% of cases at 99% percentile and improvement is 17.5% at 99% percentile when we probe Ping packets every second for a week. More experimental information can be found in [OCN].

Lower delay does not necessarily mean higher throughput. Different path segments may have different packet loss rates. Loss rate is another major factor impacting the overall TCP throughput. From some customer requirements, the target loss rate is set in the test to be less than 1% at 99% percentile and 99.9% percentile, respectively. The loss was measured between any two overlay nodes, i.e., any potential path segment. Two thousand Ping packets were sent every 20

seconds between two overlay nodes for 55 hours. This preliminary experiment showed that the packet loss rate satisfaction are 44.27% and 29.51% at the 99% and 99.9% percentiles, respectively.

Hence packet loss in an overlay segment is a key issue to be solved in such architecture. In long-haul networks, the end-to-end retransmission of lost packet can result in an extra round trip time (RTT). Such extra time is not acceptable in some latency-sensitive applications. As CION naturally consists of multiple overlay segments, LOOPS leverages this to perform local optimizations on a single hop between two overlay nodes. ("Local" here is a concept relative to end-to-end, it does not mean such optimization is limited to LAN networks.)

The following subsections present different scenarios using multiple segment-based overlay paths with a common need of local in-network loss recovery in best effort networks.

### 3.1. Tail Loss or Loss in Short Flows

When the lost segments are at the end of a transaction, TCP's fast retransmit algorithm does not work as there are no ACKs to trigger it. When a sender does not receive an ACK for a given segment within a certain amount of time called retransmission timeout (RTO), it re-sends the segment [RFC6298]. RTO can be as long as several seconds. Hence the recovery of lost segments triggered by RTO is lengthy. [I-D.dukkipati-tcpm-tcp-loss-probe] indicates that large RTOs make a significant contribution to the long tail on the latency statistics of short flows such as loading web pages.

The short flow often completes in one or two RTTs. Even when the loss is not a tail loss, it can possibly add another RTT because of end-to-end retransmission (not enough packets are in flight to trigger fast retransmit). In long-haul networks, it can result in extra time of tens or even hundreds of milliseconds.

An overlay segment transmits the aggregated flows from ON to ON. As short-lived flows are aggregated, the probability of tail loss over this specific overlay segment decreases compared to an individual flow. The overlay segment is much shorter than the end-to-end path in a Cloud- Internet overlay network, hence loss recovery over an overlay segment is faster.

### 3.2. Packet Loss in Real Time Media Streams

The Real-time transport protocol (RTP) is widely used in interactive audio and video. Packet loss degrades the quality of the received media. When the latency tolerance of the application is sufficiently

large, the RTP sender may use RTCP NACK feedback from the receiver [RFC4585] to trigger the retransmission of the lost packets before the playout time is reached at the receiver.

In a Cloud-Internet overlay network, the end-to-end path can be hundreds of milliseconds. End-to-end feedback based retransmission may be not be very useful when applications can not tolerate one more RTT of this length. Loss recovery over an overlay segment can then be used for the scenarios where RTCP NACK triggered retransmission is not appropriate.

### 3.3. Packet Loss and Congestion Control in Bulk Data Transfer

TCP congestion control algorithms such as Reno and CUBIC basically interpret packet loss as congestion experienced somewhere in the path. When a loss is detected, the congestion window will be decreased at the sender to make the sending slower. It has been observed that packet loss is not an accurate way to detect congestion in the current Internet [I-D.cardwell-iccr-g-bbr-congestion-control]. In long-haul links, when the loss is caused by non-persistent burst which is extremely short and pretty random, the sender's reaction of reducing sending rate is not able to respond in time to the instantaneous path situation or to mitigate such bursts. On the contrary, reducing window size at the sender unnecessarily or too aggressively harms the throughput for application's long lasting traffic like bulk data transfer.

The overlay nodes are distributed over the path with computing capability, they are in a better position than the end hosts to quickly deduce the underlying links' instantaneous situation from measuring the delay, loss or other metrics over the segment. Shorter round trip time over a path segment will benefit more accurate and immediate measurements for the maximum recent bandwidth available, the minimum recent latency, or trend of change. ONs can further decide if the sending rate reduction at the sender is necessary when a loss happened. Section 6.2 talks more details on this.

### 3.4. Multipathing

As an overlay path may suffer from an impairment of the underlying network, two or more overlay paths between the same set of ingress and egress overlay nodes can be combined for reliability purpose. During a transient time when a network impairment is detected, sending replicating traffic over two paths can improve reliability.

When two or more disjoint overlay paths are available as shown in Figure 3 from ON1 to ON2, different sets of traffic may use different overlay paths. For instance, one path is for low latency and the

other is for higher bandwidth, or they can be simply used as load balancing for better bandwidth utilization.

Two disjoint paths can be, for example, found by measurement to figure out the segments with very low "mathematical correlation" in latency change. When the number of overlay nodes is large, it is easy to find disjoint or partially disjoint segments. This information may be available if the ONs are managed by the network provider managing the underlying forwarding paths.

Different overlay paths may have varying characteristics, obviously. The overlay tunnel should allow the overlay path to handle the packet loss depending on its own path measurements.

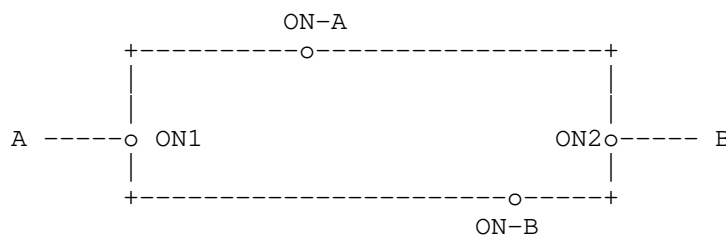


Figure 3: Example of Multiple Overlay Paths

In reference to Figure 3, both A and B are not aware of the existence of these multiple paths. A network-assistance would be valuable for the sake of better resilience and performance. Note that in a collaborative context (a.k.a., stage 2 mentioned in Section 1.2) LOOPS may target means to advertise the available path characteristics to an endpoint A/B, to allow an endpoint A/B to control the traffic distribution policy to be enforced by ON1/ON2, or to let endpoint A/B notify ON1/ON2 with their multipathing preference.

#### 4. Satellite Communication

Traditionally, satellite communications deploy PEP (performance enhancing proxy [RFC3135]) nodes around the satellite link to enhance end-to-end performance. TCP splitting is a common approach employed by such PEPs, where the TCP connection is split into three: the segment before the satellite hop, the satellite section (uplink, downlink), and the segment behind the satellite hop. This requires heavy interactions with the end-to-end transport protocols, usually without the explicit consent of the end hosts. Unfortunately, this is indistinguishable from a man-in-the-middle attack on TCP. With end-to-end encryption moving under the transport (QUIC), this approach is no longer useful.

Geosynchronous Earth Orbit (GEO) satellites have a one-way delay (up to the satellite and back) on the order of 250 milliseconds. This does not include queueing, coding and other delays in the satellite ground equipment. The Round Trip Time for a TCP or QUIC connection going over a satellite hop in both directions, in the best case, will be on the order of 600 milliseconds. And, it may be considerably longer. RTTs on this order of magnitude have significant performance implications.

Packet loss recovery is an area where splitting the TCP connection into different parts helps. Packets lost on the terrestrial links can be recovered at terrestrial latencies. Packet loss on the satellite link can be recovered more quickly by an optimized satellite protocol between the PEPs and/or link layer FEC than they could be end to end. Again, encryption makes TCP splitting no longer applicable. Enhanced error recovery at the satellite link layer helps for the loss on the satellite link but doesn't help for the terrestrial links. Even when the terrestrial segments are short, any loss must be recovered across the satellite link delay. And, there are cases when a satellite ground station connects to the general Internet with a potentially larger terrestrial segment (e.g., to a correspondent host in another country). Faster recovery over such long terrestrial segments is desirable.

Another aspect of recovery is that terrestrial loss is highly likely to be congestion related but satellite loss is more likely to be transmission errors due to link conditions. A transport endpoint slowing down because of mis-interpreting these errors as congestion losses unnecessarily reduces performance. But, at the end points, the difference between the two is not easily distinguished. To elaborate more on the loss recovery for satellite communications, while the error rate on the satellite paths is generally very low most of the time, it might get higher during special link conditions (e.g. fades due to heavy rain). The satellite hop itself does know which losses are due to link conditions as opposed to congestion, but it has no mechanism to signal this difference to the end hosts.

We will need the protocol under QUIC to try to minimize non-congestion packet drop. Specific link layers may have techniques such as satellite FEC to recover. Where the capabilities of that may be exceeded (e.g., rainfade), we can look at LOOPS-like approaches.

There are two high level classes of solutions for making encrypted transport traffic like QUIC work well over satellite:

- o Hooks in the transport protocol which can adapt to large BDPs where both the bandwidth and the latency are large. This would require end to end enhancement.

- o Capabilities (such as LOOPS) under the transport protocol to improve performance over specific segments of the path. In particular, separating the terrestrial from the satellite losses. Fixing the terrestrial loss quickly and keeping throughput high over satellite segment by not causing the end-hosts to over-reduce their sending window in case of non-congestion loss.

This document focuses on the latter.

## 5. Branch Office WAN Connection

Enterprises usually require network connections between the branch offices or between branch offices and cloud data center over geographic distances. With the increasing deployment of vCPE (virtual CPE), some services usually hosted on the CPE are moved to the provider network from the customer site. Such vCPE approach enables some value added service to be provided such as WAN optimization and traffic steering.

Figure 4 shows an example of two branch offices WAN connection via Internet. Figure 5 shows a branch office access to public cloud via a selected PoP (point of presence). vCPE connects to that PoP which can be hundreds of kilometers away via Internet. In both cases, the path segments over Internet is subject to loss. Similar problems presented in subsections of Section 3 should be solved. The GW1 may be reachable via multiple paths.

Requirements to steer traffic through different sub-paths for latency optimization, resource optimization, balancing, or other purposes are increasing. For example, directing the traffic from vCPE to a lightly loaded PoP rather than to the closest one. Mere best effort transport is not sufficient. New technologies like SFC (Service Function Chaining), SRv6 (segment routing over IPv6), and NFV/SDN used together with vCPE to enable the potentials to embed more complicated loss recovery functions at intermediate nodes in end-to-end path.

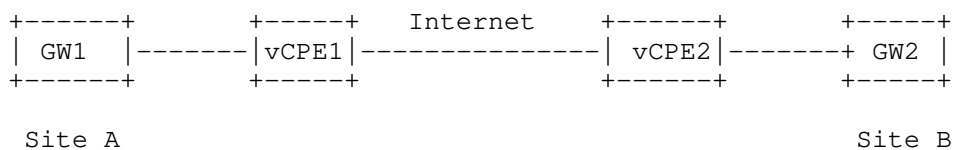


Figure 4: Branch Office WAN Connection via Internet

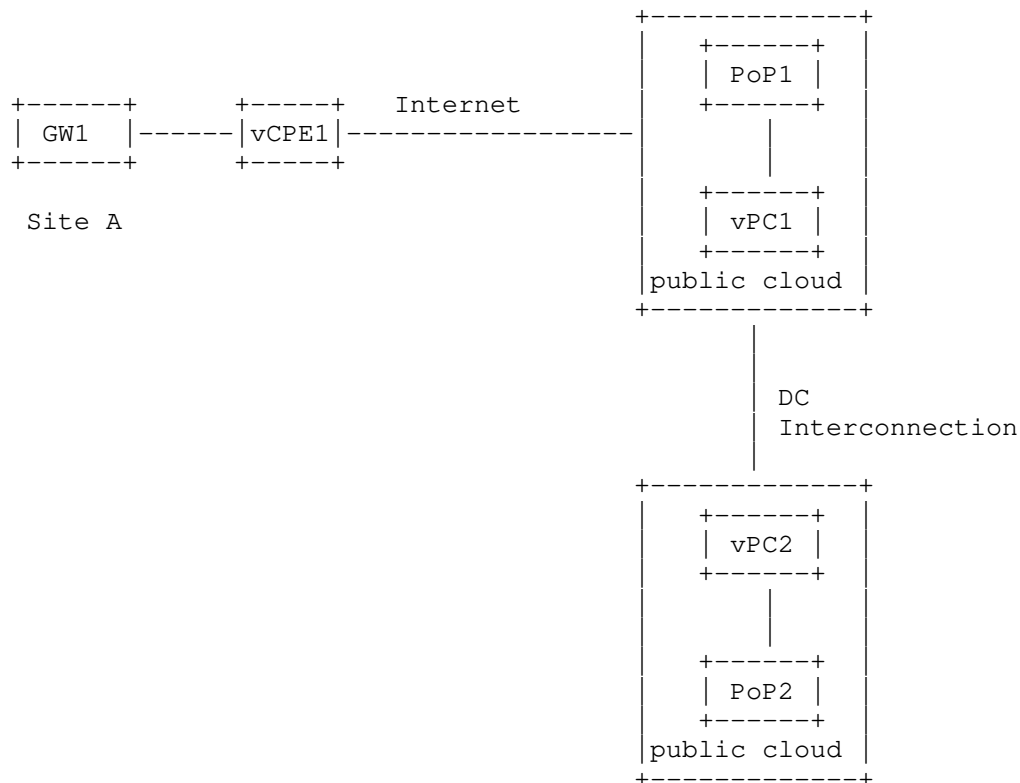


Figure 5: Enterprise Cloud Access

## 6. Features and Impacts to be Considered for LOOPS

This section provides an overview of the proposed LOOPS solution. This section is not meant to document a detailed specification, but it is meant to highlight some design choices that may be followed during the solution design phase.

LOOPS aims to improve the transport performance "locally" in addition to native end-to-end mechanism supported by a given transport protocol. This is possible because LOOPS nodes will be instantiated to partition the path into multiple segments. With the advent of automation and technologies like NFV and virtual IO, it is possible to dynamically instantiate functions to nodes. Some overlay protocols such as VXLAN [RFC7348], GENEVE [I-D.ietf-nvo3-geneve], LISP [RFC6830] or CAPWAP [RFC5415] may be used in the network. In overlay network usage scenario, LOOPS can extend a specific overlay

protocol header to perform local measurement and local recovery functions, like the example shown in Figure 6.

```
+-----+-----+-----+-----+-----+
|Outer IP hdr|Overlay hdr |LOOPS information|Inner hdr|payload  |
+-----+-----+-----+-----+-----+
```

Figure 6: LOOPS Extension Header Example

LOOPS should be designed to minimize its overhead while increasing the benefit (e.g., reduces the completion time of a video application, reduces the loss). Also, LOOPS should be designed to auto-tune itself in case its overhead is exceeding a threshold.

For example, LOOPS uses packet number space independent from that of the transport layer. Acknowledgment should be generated from ON receiver to ON sender for packet loss detection and local measurement. To reduce overhead, negative ACK over each path segment is a good choice here. A Timestamp echo mechanism, analogous to TCP's Timestamp option, should be employed in-band in LOOPS extension to measure the local RTT and variation for an overlay segment. Local in-network recovery is performed. The measurement over segment is expected to give a hint on whether the lost packet of locally recovered one was caused by congestion. Such a hint could be further feedback, using like by ECN Congestion Experienced (CE) markings, to the end host sender. It directs the end host sender if congestion window adjustment is necessary. LOOPS normally works on the overlay segment which aggregates the same type of traffic, for instance TCP traffic or finer granularity like TCP throughput sensitive traffic. LOOPS does not look into the inner packet (when an encapsulation scheme is used). Elements to be considered in LOOPS are discussed briefly here.

#### 6.1. Local Recovery and End-to-end Retransmission

There are basically two ways to perform local recovery, retransmission and FEC (Forward Error Correction). They are possibly used together in some cases. Such approaches between two overlay nodes recover the lost packet in relatively shorter distance and thus shorter latency. Therefore the local recovery is always faster compared to end-to-end.

At the same time, most transport layer protocols have their own end-to-end retransmission to recover the lost packet. It would be ideal if end-to-end retransmission at the sender was not triggered when the local recovery is successful.



End-to-end retransmission is normally triggered by a NACK as in RTCP or multiple duplicate ACKs as in TCP.

When FEC is used for local recovery, it may come with a buffer to make sure the recovered packets delivered are in order subsequently. Therefore the receiver side is unlikely to see the out-of-order packets and then send a NACK or multiple duplicate ACKs. The side effect to unnecessarily trigger end-to-end retransmit is minimum. When FEC is used, if redundancy and block size are determined, extra latency required to recover lost packets is also bounded. Then RTT variation caused by it is predictable. In some extreme case like a large number of packet loss caused by persistent burst, FEC may not be able to recover it. Then end-to-end retransmit will work as a last resort. In summary, when FEC is used as local recovery, the impact on end-to-end retransmission is limited.

When local retransmission is used, more care is required.

For packet loss in RTP streaming, local retransmission can recover those packets which would not be retransmitted end-to-end otherwise due to long RTT. It would be ideal if the retransmitted packet reaches the receiver before it sends back information that the sender would interpret as a NACK for the lost packet. Therefore when the segment(s) being retransmitted is a small portion of the whole end to end path, the retransmission will have a significant effect of improving the quality at receiver. When the sender also re-transmits the packet based on a NACK received, the receiver will receive the duplicated retransmitted packets and should ignore the duplication.

For packet loss in TCP flows, TCP RENO and CUBIC use duplicate ACKs as a loss signal to trigger the fast retransmit. There are different ways to avoid the sender's end-to-end retransmission being triggered prematurely:

- o The egress overlay node can buffer the out-of-order packets for a while, giving a limited time for a packet being retransmitted somewhere in the overlay path to reach it. The retransmitted packet and the buffered packets caused by it may increase the RTT variation at the sender. When the retransmitted latency is a small portion of RTT or the loss is rare, such RTT variation will be smoothed without much impact. Another possible way is to make the sender exclude such packets from the RTT measurement. The locally recovered packets can be specially marked and this marking is spin back to end host sender. Then RTT measurement should not use that packet.

The buffer management is nontrivial in this case. It has to be determined how many out-of-order packets can be buffered at the

egress overlay node before it gives up waiting for a successful local retransmission. In some extreme case the lost packet is not recovered successfully locally, the sender may invoke end-to-end fast retransmit slower than it would be in classic TCP.

- o If LOOPS network does not buffer the out-of-order packets caused by packet loss, TCP sender can use a time based loss detection like RACK [I-D.ietf-tcpm-rack] to prevent the TCP sender from invoking fast retransmit too early. RACK uses the notion of time to replace the conventional DUPACK threshold approach to detect losses. RACK is required to be tuned to fit the local retransmission better. If there are  $n$  similar segments over the path, segment retransmission will at least add  $RTT/n$  to the reordering window by average when the packet is lost only once over the whole overlay path. This approach is more preferred than one described in previous bullet. On the other hand, if time based loss detection is not supported at the sender, end to end retransmission will be invoked as usual. It wastes some bandwidth.

#### 6.1.1. OE to OE Measurement, Recovery, and Multipathing

When multiple segments are stitched, another type of local recovery can be performed between OE (Overlay Edge) to OE. When the segments of an overlay path have similar characteristics and/or only OE has the expected processing capability, OE to OE based local recovery can be used instead of per-segment based recovery.

If there is more than one overlay path between two OEs, multipathing can split and recombine the traffic. Measurements such as RTT and loss rate between OEs have to be specific to each path. The ingress OE can use the feedback measurement to determine the FEC parameter settings for different path. FEC can also be configured to work over the combined path. FEC should not increase redundancy over the path where a congestion is found. The egress OE should be able to remove the duplicated packets when multipathing is available.

OE to OE measurement can help each segment determine its proportion in edge to edge delay. It is useful for ON to decide if it is necessary to turn on the per segment recovery or how to fine tune the parameter settings. When the segment delay ratio is small, the segment retransmission is more effective. Such approach requires nested LOOPS function. This draft does not focus on the nest LOOPS now. More details will be discussed later if comments showing interests in it are received.

## 6.2. Congestion Control Interaction

When a TCP-like transport layer protocol is used, local recovery in LOOPS has to interact with the upper layer transport congestion control. Classic TCP adjusts the congestion window when a loss is detected and fast retransmit is invoked.

The local recovery mechanism breaks the assumption of the necessary and sufficient conditional relationship between detected packet loss and congestion control trigger at the sender in classic TCP. The loss that is locally recovered can be caused by a non-persistent congestion such as a random loss or a microburst, both of which ideally would not let the sender invoke the congestion control mechanism. But then, loss can also possibly caused by a real persistent congestion which should let the sender aware of it and reduces its sending rate.

When a local recovery takes effect, we consider the following two cases. Firstly, the classic TCP sender does not see enough number of duplicate ACKs to trigger fast retransmit. This may be due to the local recovery procedures, which hides the out-of-order packet from receiver using mechanisms like reordering buffer at egress node. Classic TCP sender in this case will not reduce congestion window as no loss is detected. Secondly, if a time based loss detection such as RACK is used, as long as the locally recovered packet's ACK reaches the sender before the reordering window expires, the congestion window will not be reduced.

Such behavior brings the desirable throughput improvement when the recovered packet is lost due to non-persistent congestion. It solves the throughput problem mentioned in Section 3.3 and Section 4. However, it also brings the risk that the sender is not able to detect a real persistent congestion in time, and then overshooting may occur. Eventually a severe congestion that is not recoverable by a local recovery mechanism will be detected by sender. In addition, it may be unfriendly to other flows (possibly pushing them out) if those flows are running over the same underlying bottleneck links.

There is a spectrum of approaches. On one end, each locally recovered packet can be treated exactly as a loss in order to invoke the congestion control at the sender to guarantee the fair sharing as classic TCP by setting its CE (Congestion Experienced) bit. Explicit Congestion Notification (ECN) can be used here as ECN marking was required to be equivalent to a packet drop [RFC3168]. Congestion control at the sender works as usual and no throughput improvement could be achieved (although the benefit of faster recovery is still there). On the other hand, ON can perform its congestion measurement over the segment, for instance local RTT and its variation trend.

Such measurement can help to determine if a lost packet by congestion. It will further decide if it is necessary to set CE marking or even what ratio is set to make the sender adjust the sending rate.

There are possible cases that the sender detects the loss even with local recovery in function. For example, when the re-ordering window in RACK is not optimally adapted, the sender may trigger the congestion control at the same time of end-to-end retransmission. If spurious retransmission detection based on DSACK [RFC3708] is used, such end-to-end retransmission will be found out unnecessary when locally recovered packets reaches the receiver successfully. Then congestion control changes will be undone at the sender. This results in similar pros and cons as described earlier. Pros are preventing the unnecessary window reduction and improving the throughput when the loss is caused by non-congestive loss. Cons are some mechanisms like ECN or its variants should be used wisely to make sure the congestion control is invoked in case of persistent congestion.

An approach where the losses on a path segment are not immediately made known to the end-to-end congestion control can be combined with a "circuit breaker" style congestion control on the path segment. When the usage of path segment by the overlay flow starts to become unfair, the path segment sends congestion signals up to the end-to-end congestion control. This must be carefully tuned to avoid unwanted oscillation.

In summary, local recovery can improve Flow Completion Time (FCT) by eliminating tail loss in small flows. As it may change loss event to out-of-order event in most cases to TCP sender, if TCP sender uses loss based congestion control, there is no much throughput improvement. We suggest ECN and spurious retransmission to be enabled when local recovery is in use, it would give the desirable throughput performance, i.e. when loss is caused by congestion, reduce congestion window; otherwise keep sender's sending rate. We do not suggest to use spurious retransmission alone together with local recovery as it may cause the TCP sender falsely undo window reduction when congestion occurs. If only ECN is enabled or neither ECN nor spurious retransmission is enabled, the throughput with local recovery in use is no much difference from that of the tradition TCP.

### 6.3. Overlay Protocol Extensions

The overlay usually has no control over how packets are routed in the underlying network between two overlay nodes, but it can control, for example, the sequence of overlay nodes a message traverses before reaching its destination. LOOPS assumes the overlay protocol can

deliver the packets in such designated sequence. Most forms of overlay networking use some sort of "encapsulation". The whole path taken can be performed by stitching multiple overlay paths, like VXLAN [RFC7348], GENEVE [I-D.ietf-nvo3-geneve], or it can be a single overlay path with a sequence of intermediate overlay nodes specified, as in SRv6 [I-D.ietf-6man-segment-routing-header]. In either way, LOOPS information is required to be embedded in some form to support the data plane measurement and feedback. Retransmission or FEC based loss recovery can be either per ON-hop or OE to OE based.

LOOPS alone has no setup requirement on control plane. Some overlay protocols, e.g., CAPWAP [RFC5415], has session setup phase, it can be used to exchange the information such as dynamic FEC parameters.

#### 6.4. Summary

LOOPS is expected to extend the existing overlay protocols in data plane. Path selection is assumed a feature provided by the overlay protocols via SDN techniques [RFC7149] or other approaches and is not a part of LOOPS. LOOPS is a set of functions to be implemented on Overlay Nodes, that will be involved in forwarding packets in a long haul overlay network. LOOPS targets the following features.

1. Local recovery: Retransmission, FEC, or combination thereof can be used as local recovery method. Such recovery mechanism is in-network. It is performed by two network nodes with computing and memory resources.
2. Local congestion measurement: Ingress/Egress overlay nodes measure the local segment RTT, loss and/or throughput to immediately get the overlay segment status.
3. Signal to end-to-end congestion control: Strategy to set ECN CE marking or simply not to recover the packet to signal the end host sender about if and/or how to adjust the sending rate is required.

#### 7. Security Considerations

LOOPS does not require access to the traffic payload in clear, so encrypted payload does not affect functionality of LOOPS.

The use of LOOPS introduces some issues which impact security. ON with LOOPS function represents a point in the network where the traffic can be potentially manipulated and intercepted by malicious nodes. Means to ensure that only legitimate nodes are involved should be considered.

Denial of service attack can be launched from an ON. A rogue ON might be able to spoof packets as if it come from a legitimate ON. It may also modify the ECN CE marking in packets to influence the sender's rate. In order to protected from such attacks, the overlay protocol itself should have some build-in security protection which inherently be used by LOOPS. The operator should use some authentication mechanism to make sure ONs are valid and non-compromised.

## 8. IANA Considerations

No IANA action is required.

## 9. Acknowledgements

Thanks to etosat mailing list about the discussion about the SatCom and LOOPS use case.

## 10. Informative References

- [RFC0793] Postel, J., "Transmission Control Protocol", STD 7, RFC 793, DOI 10.17487/RFC0793, September 1981, <<https://www.rfc-editor.org/info/rfc793>>.
- [RFC3135] Border, J., Kojo, M., Griner, J., Montenegro, G., and Z. Shelby, "Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations", RFC 3135, DOI 10.17487/RFC3135, June 2001, <<https://www.rfc-editor.org/info/rfc3135>>.
- [RFC3168] Ramakrishnan, K., Floyd, S., and D. Black, "The Addition of Explicit Congestion Notification (ECN) to IP", RFC 3168, DOI 10.17487/RFC3168, September 2001, <<https://www.rfc-editor.org/info/rfc3168>>.
- [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>>.
- [RFC3708] Blanton, E. and M. Allman, "Using TCP Duplicate Selective Acknowledgement (DSACKs) and Stream Control Transmission Protocol (SCTP) Duplicate Transmission Sequence Numbers (TSNs) to Detect Spurious Retransmissions", RFC 3708, DOI 10.17487/RFC3708, February 2004, <<https://www.rfc-editor.org/info/rfc3708>>.

- [RFC4585] Ott, J., Wenger, S., Sato, N., Burmeister, C., and J. Rey, "Extended RTP Profile for Real-time Transport Control Protocol (RTCP)-Based Feedback (RTP/AVPF)", RFC 4585, DOI 10.17487/RFC4585, July 2006, <<https://www.rfc-editor.org/info/rfc4585>>.
- [RFC4588] Rey, J., Leon, D., Miyazaki, A., Varsa, V., and R. Hakenberg, "RTP Retransmission Payload Format", RFC 4588, DOI 10.17487/RFC4588, July 2006, <<https://www.rfc-editor.org/info/rfc4588>>.
- [RFC5415] Calhoun, P., Ed., Montemurro, M., Ed., and D. Stanley, Ed., "Control And Provisioning of Wireless Access Points (CAPWAP) Protocol Specification", RFC 5415, DOI 10.17487/RFC5415, March 2009, <<https://www.rfc-editor.org/info/rfc5415>>.
- [RFC5681] Allman, M., Paxson, V., and E. Blanton, "TCP Congestion Control", RFC 5681, DOI 10.17487/RFC5681, September 2009, <<https://www.rfc-editor.org/info/rfc5681>>.
- [RFC6298] Paxson, V., Allman, M., Chu, J., and M. Sargent, "Computing TCP's Retransmission Timer", RFC 6298, DOI 10.17487/RFC6298, June 2011, <<https://www.rfc-editor.org/info/rfc6298>>.
- [RFC6830] Farinacci, D., Fuller, V., Meyer, D., and D. Lewis, "The Locator/ID Separation Protocol (LISP)", RFC 6830, DOI 10.17487/RFC6830, January 2013, <<https://www.rfc-editor.org/info/rfc6830>>.
- [RFC7149] Boucadair, M. and C. Jacquenet, "Software-Defined Networking: A Perspective from within a Service Provider Environment", RFC 7149, DOI 10.17487/RFC7149, March 2014, <<https://www.rfc-editor.org/info/rfc7149>>.
- [RFC7348] Mahalingam, M., Dutt, D., Duda, K., Agarwal, P., Kreeger, L., Sridhar, T., Bursell, M., and C. Wright, "Virtual eXtensible Local Area Network (VXLAN): A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks", RFC 7348, DOI 10.17487/RFC7348, August 2014, <<https://www.rfc-editor.org/info/rfc7348>>.
- [RFC8517] Dolson, D., Ed., Snellman, J., Boucadair, M., Ed., and C. Jacquenet, "An Inventory of Transport-Centric Functions Provided by Middleboxes: An Operator Perspective", RFC 8517, DOI 10.17487/RFC8517, February 2019, <<https://www.rfc-editor.org/info/rfc8517>>.

- [I-D.dukkipati-tcpm-tcp-loss-probe]  
Dukkipati, N., Cardwell, N., Cheng, Y., and M. Mathis,  
"Tail Loss Probe (TLP): An Algorithm for Fast Recovery of  
Tail Losses", draft-dukkipati-tcpm-tcp-loss-probe-01 (work  
in progress), February 2013.
- [I-D.ietf-nvo3-geneve]  
Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic  
Network Virtualization Encapsulation", draft-ietf-  
nvo3-geneve-13 (work in progress), March 2019.
- [I-D.ietf-tcpm-rack]  
Cheng, Y., Cardwell, N., Dukkipati, N., and P. Jha, "RACK:  
a time-based fast loss detection algorithm for TCP",  
draft-ietf-tcpm-rack-05 (work in progress), April 2019.
- [I-D.ietf-6man-segment-routing-header]  
Filsfils, C., Dukes, D., Previdi, S., Leddy, J.,  
Matsushima, S., and d. daniel.voyer@bell.ca, "IPv6 Segment  
Routing Header (SRH)", draft-ietf-6man-segment-routing-  
header-21 (work in progress), June 2019.
- [I-D.cardwell-iccr-g-bbr-congestion-control]  
Cardwell, N., Cheng, Y., Yeganeh, S., and V. Jacobson,  
"BBR Congestion Control", draft-cardwell-iccr-g-bbr-  
congestion-control-00 (work in progress), July 2017.
- [DOI\_10.1109\_ICDCS.2016.49]  
Cai, C., Le, F., Sun, X., Xie, G., Jamjoom, H., and R.  
Campbell, "CRONets: Cloud-Routed Overlay Networks", 2016  
IEEE 36th International Conference on Distributed  
Computing Systems (ICDCS), DOI 10.1109/icdcs.2016.49, June  
2016.
- [DOI\_10.1145\_3038912.3052560]  
Haq, O., Raja, M., and F. Dogar, "Measuring and Improving  
the Reliability of Wide-Area Cloud Paths", Proceedings of  
the 26th International Conference on World Wide Web -  
WWW '17, DOI 10.1145/3038912.3052560, 2017.
- [OCN]  
Xu, Z., Ju, R., Gu, L., Wang, W., Li, J., Li, F., and L.  
Han, "Using Overlay Cloud Network to Accelerate Global  
Communications", INFOCOM ICCN 2019, April 2019,  
<[https://github.com/zhaoguixu/INFOCOM19\\_ICCN/blob/master/  
ocn.pdf](https://github.com/zhaoguixu/INFOCOM19_ICCN/blob/master/ocn.pdf)>.



Authors' Addresses

Yizhou Li  
Huawei Technologies  
101 Software Avenue,  
Nanjing 210012  
China

Phone: +86-25-56624584  
Email: liyizhou@huawei.com

Xingwang Zhou  
Huawei Technologies  
101 Software Avenue,  
Nanjing 210012  
China

Email: zhouxingwang@huawei.com

Mohamed Boucadair  
Orange

Email: mohamed.boucadair@orange.com

Jianglong Wang  
China Telecom

Email: wangjl1.bri@chinatelecom.cn

TSVWG  
Internet-Draft  
Intended status: Standards Track  
Expires: November 28, 2019

M. Welzl  
University of Oslo  
C. Bormann, Ed.  
Universitaet Bremen TZI  
May 27, 2019

LOOPS Generic Information Set  
draft-welzl-loops-gen-info-00

Abstract

LOOPS (Local Optimizations on Path Segments) aims to provide local (not end-to-end but in-network) recovery of lost packets to achieve better data delivery in the presence of losses.

[I-D.li-tsvwg-loops-problem-opportunities] provides an overview over the problems and optimization opportunities that LOOPS could address.

The present document is a strawman for the set of information that would be interchanged in a LOOPS protocol, without already defining a specific data packet format.

The generic information set needs to be mapped to a specific encapsulation protocol to actually run the LOOPS optimizations. The current version of this document contains sketches of bindings to GUE [I-D.ietf-intarea-gue] and Geneve [I-D.ietf-nvo3-geneve].

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 November 28, 2019.

## 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. Terminology . . . . .	5
2. Challenges . . . . .	6
2.1. No Access to End-to-End Transport Information . . . . .	6
2.2. Path Asymmetry . . . . .	6
2.3. Reordering vs. Spurious Retransmission . . . . .	6
2.4. Informing the End-to-End Transport . . . . .	7
2.5. Congestion Detection . . . . .	8
3. Simplifying assumptions . . . . .	8
4. LOOPS Generic Information Set . . . . .	9
4.1. Setup Information . . . . .	9
4.2. Forward Information . . . . .	9
4.3. Reverse Information . . . . .	10
5. LOOPS General Operation . . . . .	11
5.1. Initial Packet Sequence Number . . . . .	11
5.2. Acknowledgement Generation . . . . .	11
5.3. Measurement . . . . .	12
5.4. Loss detection and Recovery . . . . .	12
5.4.1. Local Retransmission . . . . .	12
5.4.2. FEC . . . . .	13
5.5. Discussion . . . . .	13
6. Sketches of Bindings to Tunnel Protocols . . . . .	13
6.1. Embedding LOOPS in Geneve . . . . .	14
6.2. Embedding LOOPS in GUE . . . . .	14
7. IANA Considerations . . . . .	14
8. Security Considerations . . . . .	15
9. Informative References . . . . .	15
Appendix A. Protocol used in Prototype Implementation . . . . .	16
A.1. Block Code FEC . . . . .	17
Appendix B. Transparent mode . . . . .	18
B.1. Packet identification . . . . .	19

B.2. Generic information and protocol operation . . . . .	20
Acknowledgements . . . . .	20
Authors' Addresses . . . . .	20

## 1. Introduction

Today's networks exhibit a wide variety of data rates and, relative to those, processing power and memory capacities of nodes acting as routers. For instance, networks that employ tunneling to build overlay networks may position powerful virtual router nodes in the network to act as tunnel endpoints. The capabilities available in the more powerful cases provide new opportunities for optimizations.

LOOPS (Local Optimizations on Path Segments) aims to provide local (not end-to-end but in-network) recovery of lost packets to achieve better data delivery. [I-D.li-tsvwg-loops-problem-opportunities] provides an overview over the problems and optimization opportunities that LOOPS could address. One simplifying assumption (Section 3) in the present document is that LOOPS segments operate independently from each other, each as a pair of a LOOPS Ingress and a LOOPS Egress node.

The present document is a strawman for the set of information that would be interchanged in a LOOPS protocol between these nodes, without already defining a specific data packet format. The main body of the document defines a mode of the LOOPS protocol that is based on traditional tunneling, the "tunnel mode". Appendix B is an even rougher strawman of a radically different, alternative mode that we call "transparent mode". These different modes may be applicable to different usage scenarios and will be developed in parallel, with a view of ultimately standardizing one or both of them.

For tunnel mode, the generic information set needs to be mapped to a specific encapsulation protocol to actually run the LOOPS optimizations. LOOPS is not tied to any specific overlay protocol, but is meant to run embedded into a variety of tunnel protocols. LOOPS information is added as part of a tunnel protocol header at the LOOPS ingress as shown in Figure 1. The current version of this document contains sketches of bindings to GUE [I-D.ietf-intarea-gue] and Geneve [I-D.ietf-nvo3-geneve].

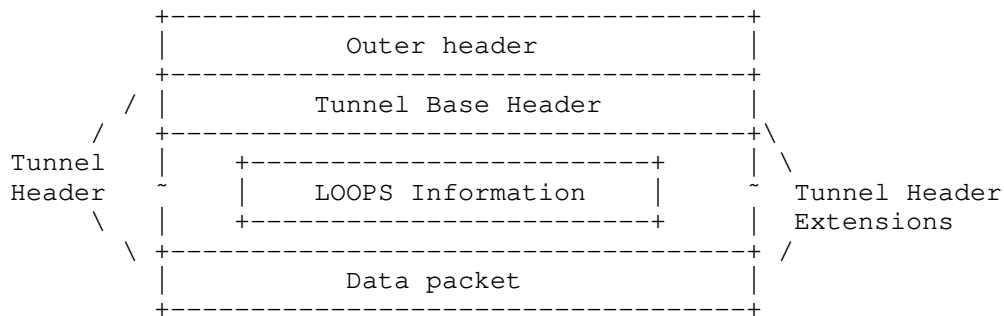


Figure 1: Packet in Tunnel with LOOPS Information

Figure 2 is extracted from the LOOPS problems and opportunities document [I-D.li-tsvwg-loops-problem-opportunities]. It illustrates the basic architecture and terms of the applicable scenario of LOOPS. Not all of the concepts introduced in the problems and opportunities document are actually used in the current strawman specification; Section 3 lays out some simplifying assumptions that the present proposal makes.

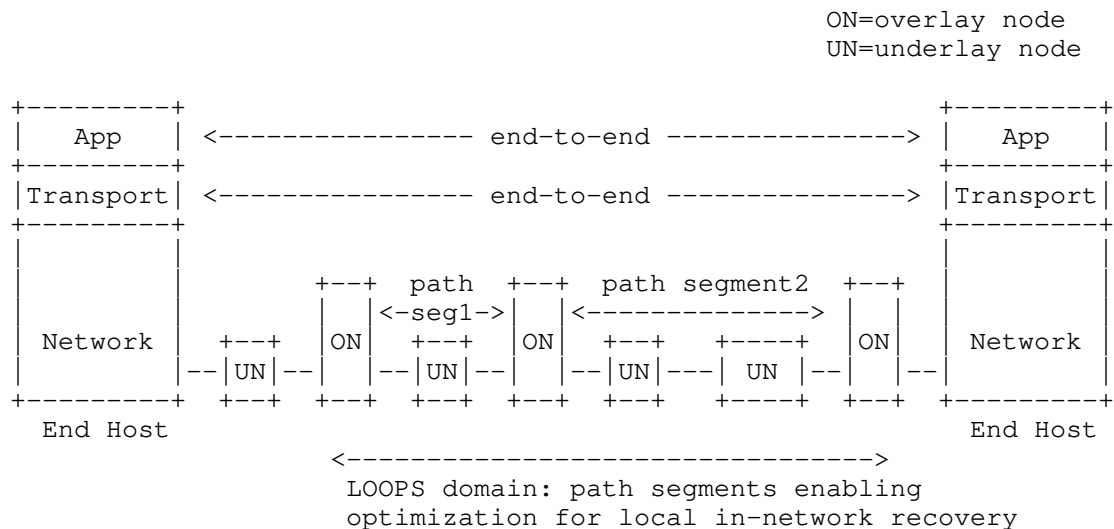


Figure 2: LOOPS Usage Scenario

### 1.1. Terminology

This document makes use of the terminology defined in [I-D.li-tsvwg-loops-problem-opportunities]. This section defines additional terminology used by this document.

**Data packets:** The payload packets that enter and exit a LOOPS segment.

**LOOPS Segment:** A part of an end-to-end path covered by a single instance of the LOOPS protocol, the sub-path between the LOOPS Ingress and the LOOPS Egress.

**LOOPS Ingress:** The node that forwards data packets and forward information into the LOOPS segment, potentially performing retransmission and forward error correction based on acknowledgements and measurements received from the LOOPS Egress.

**LOOPS Egress:** The node that receives the data packets and forward information from the LOOPS ingress, sends acknowledgements and measurements back to the LOOPS ingress (reverse information), potentially recovers data packets from forward error correction information received.

**LOOPS Nodes:** Collective term for LOOPS Ingress and LOOPS Egress in a LOOPS Segment.

**Forward Information:** Information that is added to the stream of data packets in the forward direction by the LOOPS Ingress.

**Reverse Information:** Information that flows in the reverse direction, from the LOOPS Egress back to the LOOPS Ingress.

**Setup Information:** Information that is not transferred as part of the Forward or Reverse Information, but is part of the setup of the LOOPS Nodes.

**PSN:** Packet Sequence Number, a sequence number identifying a data packet.

**Sender:** Original sender of a packet on an end-to-end path that includes one or more LOOPS segment(s).

**Receiver:** Ultimate receiver of a packet on an end-to-end path that includes one or more LOOPS segment(s).

## 2. Challenges

LOOPS has to perform well in the presence of some challenges, which are discussed in this section.

### 2.1. No Access to End-to-End Transport Information

LOOPS is defined to be independent of the content of the packets being forwarded: there is no dependency on transport-layer or higher information. The intention is to keep LOOPS useful with a traffic mix that may contain encrypted transport protocols such as QUIC as well as encrypted VPN traffic.

### 2.2. Path Asymmetry

A LOOPS segment is defined as a unidirectional forwarding path. The tunnel might be shared with a LOOPS segment in the inverse direction; this then allows to piggyback Reverse Information on encapsulated packets on that segment. But there is no guarantee that the inverse direction of any end-to-end-path crosses that segment, so the LOOPS optimizations have to be useful on their own in each direction.

### 2.3. Reordering vs. Spurious Retransmission

The end-to-end transport layer protocol may have its own retransmission mechanism to recover lost packets. When LOOPS recovers a loss, ideally this local recovery would avoid the triggering of a retransmission at the end-to-end sender.

Whether this is possible depends on the specific end-to-end mechanism used for triggering retransmission. When end-to-end retransmission is triggered by receiving a sequence of duplicate acknowledgements (DUPACKs), and with more than a few packets in flight, the recovered packet is likely to be too late to fill the hole in the sequence number space that triggers the DUPACK detection.

(Given a reasonable setting of parameters, the local retransmission will still arrive earlier than the end-to-end retransmission and will possibly unblock application processing earlier; with spurious retransmission detection, there also will be little long-term effect on the send rate.)

The waste of bandwidth caused by a DUPACK-based end-to-end retransmission can be avoided when the end-to-end loss detection is based on time instead of sequence numbers, e.g., with RACK [I-D.ietf-tcpm-rack]. This requires a limit on the additional latency that LOOPS will incur in its attempt to recover the loss locally. In the present version of this document, opportunity to set

such a limit is provided in the Setup Information. The limit can be used to compute a deadline for retransmission, but also can be used to choose FEC parameters that keep extra latency low.

#### 2.4. Informing the End-to-End Transport

Congestion control at the end-to-end sender is used to adapt its sending rate to the network congestion status. In typical TCP senders, packet loss implies congestion and leads to a reduction in sending rate. With LOOPS operating, packet loss can be masked from the sender as the loss may have been locally recovered. In this case, rate reduction may not be invoked at the sender. This is a desirable performance improvement if the loss was a random loss.

If LOOPS successfully conceals congestion losses from the end-to-end transport protocol, that might increase the rate to a level that congests the LOOPS segment, or that causes excessive queueing at the LOOPS ingress. What LOOPS should be able to achieve is to let the end host sender invoke the rate reduction mechanism when there is a congestion loss no matter if the lost packet was recovered locally.

As with any tunneling protocol, information about congestion events inside the tunnel needs to be exported to the end-to-end path the tunnel is part of. See e.g., [RFC6040] for a discussion of how to do this in the presence of ECN. A more recent draft, [I-D.ietf-tsvwg-tunnel-congestion-feedback], proposes to activate ECN for the tunnel regardless of whether the end-to-end protocol signals the use of an ECN-capable transport (ECT), which requires more complicated action at the tunnel egress.

A sender that interprets reordering as a signal of packet loss (DUPACKs) initiates a retransmission and reduces the sending rate. When spurious retransmission detection (e.g., via F-RTO [RFC5862] or DSACK [RFC3708] is enabled by the TCP sender, it will often be able to undo the unnecessary window reduction. As LOOPS recovers lost packets locally, in most cases the end host sender will eventually find out its reordering-based retransmission (if any) is spurious. This is an appropriate performance improvement if the loss was a random loss. For congestion losses, a congestion event needs to be signaled to the end-to-end transport.

If the end-to-end transport is ECN-capable (which is visible at the IP level), congestion loss events can easily be signaled to them by setting the CE (congestion experienced) mark. If LOOPS detects a congestion loss for a non-ECT packet, it needs to signal a congestion loss event by introducing a packet loss. This can be done by choosing not to retransmit or repair the packet loss locally in this case. Note that one congestion loss per end-to-end RTT is sufficient



to provide the rate reduction, so LOOPS may still be able to recover most packets, in particular for burst losses. (As LOOPS does not interact with the end-to-end transport, it does not know the end-to-end RTT. Some lower bound derived from configuration and measurements could be used instead.)

### 2.5. Congestion Detection

Properly informing the end-to-end transport protocol about congestion loss events requires distinguishing these from random losses. In some special cases, distinguishing information may be available from a link layer (e.g., see Section 3 of [I-D.li-tsvwg-loops-problem-opportunities]). By enabling ECN inside the tunnel, congestion events experienced at ECN-capable routers will usually be identified by the CE mark, which clearly rules out a random loss.

In the general case, the segment may be composed of hops without such special indications. In these cases, some detection mechanism is required to provide this distinguishing information. The specific mechanism used by an implementation is out of scope of LOOPS, but LOOPS will need to provide measurement information for this mechanism. For instance, congestion detection might be based on path segment latency information, the proper measurement of which therefore requires special attention in LOOPS.

### 3. Simplifying assumptions

The above notwithstanding, Implementations may want to make use of indicators such as transport layer port numbers to partition a tunnel flow into separate application flows, e.g., for active queue management (AQM). Any such functionality is orthogonal to the LOOPS protocol itself and thus out of scope for the present document.

One observation that simplifies the design of LOOPS in comparison to that of a reliable transport protocol is that LOOPS does not have to recover every packet loss. Therefore, probabilistic approaches, and simply giving up after some time has elapsed, can simplify the protocol significantly.

For now, we assume that LOOPS segments that may line up on an end-to-end path operate independently of each other. Since the objective of LOOPS ultimately is to assist the end-to-end protocol, it is likely that some cooperation between them would be beneficial, e.g., to obtain some measurements that cover a larger part of the end-to-end path. For instance, cooperating LOOPS segments could try to divide up permissible increases to end-to-end latency between them. This is out of scope for the present version.

Another simplifying assumption is that LOOPS nodes have reasonably precise absolute time available to them, so there is no need to burden the LOOPS protocol with time synchronization. How this is achieved is out of scope.

LOOPS nodes are created and set up (information about their peers, parameters) by some control plane mechanism that is out of scope for this specification. This means there is no need in the LOOPS protocol itself to manage setup information.

#### 4. LOOPS Generic Information Set

This section sketches a generic information set for the LOOPS protocol. Entries marked with (\*) are items that may not be necessary and probably should be left out of an initial specification.

##### 4.1. Setup Information

Setup Information might include:

- o encapsulation protocol in use, and its vital parameters
- o identity of LOOPS ingress and LOOPS egress; information relevant for running the encapsulation protocol such as port numbers
- o target maximum latency increase caused by the operation of LOOPS on this segment
- o maximum retransmission count (\*)

In the data plane, we have forward information (information added to each data packet) and reverse information. The latter can be sent in separate packets (e.g., Geneve control-only packets [I-D.ietf-nvo3-geneve]) and/or piggybacked like the forward information.

##### 4.2. Forward Information

In the forward information, we have identified:

- o tunnel type (a few bits, meaning agreed between Ingress and Egress)
- o packet sequence number PSN (20+ bits), counting the LOOPS packets transmitted by the LOOPS ingress (i.e., retransmissions receive a new PSN)

- o an "ACK desirable" flag (one bit, usually set for a certain percentage of the data packets only)
- o anything that the FEC scheme needs.

The first four together (say, 3+24+4+1) might even fit into 32 bits, but probably need up to 48 bits total. FEC info of course often needs more space.

(Note that in this proposal there is no timestamp in the forward information; see Section 5.3.)

24 bits of PSN, minus one bit for sequence number arithmetic, gives 8 million packets (or 2.4 GB at typical packet sizes) per worst-case RTT. So if that is, say, 30 seconds, this would be enough to fill 640 Mbit/s.

#### 4.3. Reverse Information

For the reverse information, we have identified:

- o one optional block 1, possibly repeated:
- o PSN being acknowledged
- o absolute time of reception for the packet acknowledged (PSN)
- o one optional block 2, possibly repeated:
- o an ACK bitmap (based on PSN), always starting at a multiple of 8
- o a delta indicating the end PSN of the bitmap (actually the first PSN that is beyond it), using  $(\text{Acked-PSN} \& \sim 7) + 8 * (\text{delta} + 1)$  as the end of the bitmap. Acked-PSN in that formula is the previous block 1 PSN seen in this packet, or 0 if none so far.

Block 1 and Block 2 can be interspersed and repeated. They can be piggybacked on a reverse direction data packet or sent separately if none occurs within some timeout. They will usually be aggregated in some useful form. Block 1 information sets are only returned for packets that have "ACK desirable" set. Block 2 information is sent by the receiver based on some saturation scheme (e.g., at least three copies for each PSN span over time). Still, it might be possible to go down to 1 or 2 amortized bytes per forward packet spent for all this.

The latency calculation is done by the sender, who occasionally sets "ACK desirable", and notes down the absolute time of transmission for

this data packet (the timekeeping can be done quite efficiently as deltas). Upon reception of a block 1 ACK, it can then subtract that from the absolute time of reception indicated. This assumes time synchronization between the nodes is at least as good as the precision of latency measurement needed, which should be no problem with IEEE 1588 PTP synchronization (but could be if using NTP-based synchronization only). A sender can freely garbage collect noted down transmission time information; doing this too early just means that the quality of the RTT sampling will reduce.

## 5. LOOPS General Operation

In the Tunnel Mode described in the main body of this document, LOOPS information is carried by some tunnel encapsulation.

### 5.1. Initial Packet Sequence Number

There is no connection establishment procedure in LOOPS. The initial PSN is assigned unilaterally by the LOOPS Ingress.

Because of the short time that is usually set in the maximum latency increase, there is little damage from a collision of PSNs with packets still in flight from previous instances of LOOPS.

Collisions can be minimized by assigning initial PSNs randomly, or using stable storage. Random assignment is more useful for longer PSNs, where the likelihood of overlap will be low. The specific way a LOOPS ingress uses stable storage is a local matter and thus out of scope. (Implementation note: this can be made to work similar to secure nonce generation with write attenuation: Say, every 10000 packets, the sender notes down the PSN into stable storage. After a reboot, it reloads the PSN and adds 10000 in sequence number arithmetic [RFC1982], plus maybe another 10000 so the sender does not have to wait for the store operation to succeed before sending more packets.)

### 5.2. Acknowledgement Generation

A data packet forwarded by the LOOPS ingress always carries PSN information. The LOOPS egress uses the largest newly received PSN with the "ACK desired" bit as the ACK number in the block 1 part of the acknowledgement. This means that the LOOPS ingress gets to modulate the number of acknowledgement sent by the LOOPS egress. However, whenever an out-of-order packet arrives while there still are "holes" in the PSNs received, the LOOPS receiver should generate a block 2 acknowledgement immediately that the LOOPS sender can use as a NACK list.

Reverse information can be piggybacked in a reverse direction data packet. When the reverse direction has no user data to be sent, a pure reverse information packet needs to be generated. This may be based on a short delay during which the LOOPS egress waits for a data packet to piggyback on. (To reduce MTU considerations, the egress could wait for less-than-full data packets.)

### 5.3. Measurement

When sending a block 1 acknowledgement, the LOOPS egress indicates the absolute time of reception of the packet. The LOOPS ingress can subtract the absolute time of transmission that it still has available, resulting in one high quality latency sample. (In an alternative design, the forward information could include the absolute time of transmission as well, and block1 information would echo it back. This trades memory management at the ingress for increased bandwidth and MTU reduction.)

The LOOPS ingress can also use the time of reception of the block 1 acknowledgement to obtain a segment RTT sample. Note that this will include any wait time the LOOPS egress incurs while waiting for a piggybacking opportunity -- this is appropriate, as all uses of an RTT will be for keeping a retransmission timeout.

To maintain quality of information during idle times, the LOOPS ingress may send keepalive packets, which are discarded at the LOOPS egress after sending acknowledgements. The indication that a packet is a keepalive packet is dependent on the encapsulation protocol.

### 5.4. Loss detection and Recovery

There are two ways for LOOPS local recovery, retransmission and FEC.

#### 5.4.1. Local Retransmission

When retransmission is used as recovery mechanism, the LOOPS ingress detects a packet loss by receiving a NACK or by local timeout (using a RTO value that might be calculated as in [RFC6298]). It might employ a DUPACK-like or a RACK-like mechanism for delayed reaction to a NACK.

When a retransmission is desired (see Section 2.4 for why it might not be), the LOOPS ingress performs the local in-network recovery by retransmitting the packet. Further retransmissions may be desirable if the NACK is persistent beyond an RTO, as long as the maximum latency increase is not reached.

#### 5.4.2. FEC

FEC is another way to perform local recovery. When FEC is in use, a FEC header is sent with data packets as well as with special repair packets added to the flow. The specific FEC scheme used could be defined in the Setup Information, using a mechanism like [RFC5052]. The FEC rate (amount of redundancy added) and possibly the FEC scheme could be unilaterally adjusted by the LOOPS ingress in an adaptive mechanism based on the measurement information.

#### 5.5. Discussion

Without progress in the way that end-host transport protocols handle reordering, LOOPS will be unable to prevent end-to-end retransmissions that duplicate effort that is spent in local retransmissions. It depends on parameters of the path segment whether this wasted effort is significant or not.

One remedy against this waste could be the introduction of resequencing at the LOOPS Egress node. This increases overall mean packet latency, but does not always increase actual end-to-end data stream latency if a head-of-line blocking transport such as TCP is in use. For applications with a large percentage of legacy TCP end-hosts and sufficient processing capabilities at the LOOPS Egress node, resequencing may be a viable choice. Note that resequencing could be switched off and on depending on some measurement information.

To enable resequencing at the LOOPS Egress, a packet numbering scheme is needed that allows the LOOPS Egress to reconstruct the sequence at the LOOPS ingress. This could be done by reverting to a traditional packet sequence number counting incoming data packets, possibly combined with a "retransmission" bit that indicates that the specific LOOPS packet is a retransmission and not the original transmission. (The acknowledgement/measurement ambiguity could be further reduced by adding transmission counter TC that counts transmission/retransmission for this PSN; a few bits should be enough for the limited retransmission envisaged.)

#### 6. Sketches of Bindings to Tunnel Protocols

The LOOPS information defined above in a generic way can be mapped to specific tunnel encapsulation protocols. Sketches for two tunnel protocols are given below: Geneve (Section 6.1), and GUE (Section 6.2).

### 6.1. Embedding LOOPS in Geneve

Geneve [I-D.ietf-nvo3-geneve] is an extensible overlay protocol which can embed LOOPS functions. Geneve uses TLVs to carry optional information between NVEs. NVE is logically the same entity as the LOOPS node.

For Geneve, a new LOOPS TLV needs to be defined and its format needs to be consistent with LOOPS generic information in Section 4. When the Geneve LOOPS TLV is put in forward information, NVEs should be able to process it. Any settings needed can be provided in the Setup Information.

In the reverse direction, when no data packets are available for piggybacking, a control only packet will be used to carry the LOOPS reverse information. Such a control only packet sets the 'O' bit in the Geneve header and has no real user data.

VNI is a mandatory field in Geneve base header. The LOOPS TLV should function on the tunnel between two NVEs without looking at the VNI value. The LOOPS PSN number space is local to the overlay tunnel regardless of the VNI inside. At the ingress NVE, there are different ways to decide whether a packet should go to LOOPS enabled tunnel, e.g. by protocol number (TCP/UDP certain ports) or by VNI.

### 6.2. Embedding LOOPS in GUE

GUE [I-D.ietf-intarea-gue] is an extensible overlay protocol which can embed LOOPS functions. GUE uses flags to indicate the presence of fixed length header extensions. It also allows variable length extensions to be put in "Private data" field. A new LOOPS data block in the "private data" field needs to be defined based on the LOOPS generic information in Section 4.

In the reverse direction, when no data packets are available for piggybacking, LOOPS reverse information is carried in a control message with the C-bit set in the GUE header. The Proto/ctype field contains a control message type when C bit is set. Hence a new control message type should be defined for such LOOPS reverse information.

## 7. IANA Considerations

No IANA action is required at this stage. When a LOOPS representation is designed for a specific tunneling protocol, new codepoints will be required in the registries that pertain to that protocol.

## 8. Security Considerations

To be defined.

## 9. Informative References

- [RFC1982] Elz, R. and R. Bush, "Serial Number Arithmetic", RFC 1982, DOI 10.17487/RFC1982, August 1996, <<https://www.rfc-editor.org/info/rfc1982>>.
- [RFC3708] Blanton, E. and M. Allman, "Using TCP Duplicate Selective Acknowledgement (DSACKs) and Stream Control Transmission Protocol (SCTP) Duplicate Transmission Sequence Numbers (TSNs) to Detect Spurious Retransmissions", RFC 3708, DOI 10.17487/RFC3708, February 2004, <<https://www.rfc-editor.org/info/rfc3708>>.
- [RFC5052] Watson, M., Luby, M., and L. Vicisano, "Forward Error Correction (FEC) Building Block", RFC 5052, DOI 10.17487/RFC5052, August 2007, <<https://www.rfc-editor.org/info/rfc5052>>.
- [RFC5862] Yasukawa, S. and A. Farrel, "Path Computation Clients (PCC) - Path Computation Element (PCE) Requirements for Point-to-Multipoint MPLS-TE", RFC 5862, DOI 10.17487/RFC5862, June 2010, <<https://www.rfc-editor.org/info/rfc5862>>.
- [RFC6040] Briscoe, B., "Tunnelling of Explicit Congestion Notification", RFC 6040, DOI 10.17487/RFC6040, November 2010, <<https://www.rfc-editor.org/info/rfc6040>>.
- [RFC6298] Paxson, V., Allman, M., Chu, J., and M. Sargent, "Computing TCP's Retransmission Timer", RFC 6298, DOI 10.17487/RFC6298, June 2011, <<https://www.rfc-editor.org/info/rfc6298>>.
- [RFC6330] Luby, M., Shokrollahi, A., Watson, M., Stockhammer, T., and L. Minder, "RaptorQ Forward Error Correction Scheme for Object Delivery", RFC 6330, DOI 10.17487/RFC6330, August 2011, <<https://www.rfc-editor.org/info/rfc6330>>.
- [I-D.ietf-tcpm-rack] Cheng, Y., Cardwell, N., Dukkupati, N., and P. Jha, "RACK: a time-based fast loss detection algorithm for TCP", draft-ietf-tcpm-rack-05 (work in progress), April 2019.



[I-D.ietf-nvo3-geneve]

Gross, J., Ganga, I., and T. Sridhar, "Geneve: Generic Network Virtualization Encapsulation", draft-ietf-nvo3-geneve-13 (work in progress), March 2019.

[I-D.ietf-intarea-gue]

Herbert, T., Yong, L., and O. Zia, "Generic UDP Encapsulation", draft-ietf-intarea-gue-07 (work in progress), March 2019.

[I-D.li-tsvwg-loops-problem-opportunities]

Yizhou, L. and X. Zhou, "LOOPS (Localized Optimizations of Path Segments) Problem Statement and Opportunities", draft-li-tsvwg-loops-problem-opportunities-02 (work in progress), May 2019.

[I-D.ietf-tsvwg-tunnel-congestion-feedback]

Wei, X., Yizhou, L., Boutros, S., and L. Geng, "Tunnel Congestion Feedback", draft-ietf-tsvwg-tunnel-congestion-feedback-07 (work in progress), May 2019.

## Appendix A. Protocol used in Prototype Implementation

This appendix describes, in a somewhat abstracted form, the protocol as used in a prototype implementation, as described by Yizhou Li, and Xingwang Zhou.

The prototype protocol can be run in one of two modes (defined by preconfiguration):

- o Retransmission mode
- o Forward Error Correction (FEC) mode

Forward information is piggybacked in data packets.

Reverse information can be carried in a pure acknowledgement packet or piggybacked when carrying packets for the inverse direction.

The forward information includes:

- o Packet Sequence Number (PSN) (32 bits): This identifies a packet over a specific overlay segment from a specific LOOPS Ingress. If a packet is retransmitted by LOOPS, the retransmission uses the original PSN.
- o Timestamp (32 bits): Information, in a format local to the LOOPS ingress, that provides the time when the packet was sent. In the

current implementation, a 32-bit unsigned value specifying the time delta in some granularity from the epoch time to the sending time of the packet carrying this timestamp. The granularity can be from 1 ms to 1 second. The epoch time follows the current TCP practice which is 1 January 1970 00:00:00 UTC. Note that a retransmitted packet uses its own Timestamp.

- o FEC Info for Block Code (56 bits): This header is used in FEC mode. It currently only provides for a block code FEC scheme. It includes the Source Block Number (SBN), Encoding Symbol ID (ESI), number of symbols in a single source block and symbol size. Appendix A.1 gives more details on FEC.

The reverse information includes:

- o ACK Number (32 bits): The largest (in sequence number arithmetic [RFC1982]) PSN received so far.
- o NACK List (variable): This indicates an array of PSN numbers to describe the PSN "holes" preceding the ACK number. It conceptually lists the PSNs of every packet perceived as lost by the LOOPS egress. In actual use, it is truncated.
- o Echoed Timestamp (32 bits): The timestamp received with the packet being acknowledged.

#### A.1. Block Code FEC

The prototype currently uses a block code FEC scheme (RaptorQ [RFC6330]). The fields in the FEC Info forward information are:

- o Source Block Number (SBN): 16 bits. An integer identifier for the source block that the encoding symbols within the packet relate to.
- o Encoding Symbol ID (ESI): 16 bits. An integer identifier for the encoding symbols within the packet.
- o K: 8 bits. Number of symbols in a single source block.
- o T: 16 bits. Symbol size in bytes.

The LOOPS Ingress uses the data packet in Figure 1 to generate the encoding packet. Both source packets and repair packets carry the FEC header information; the LOOPS Egress reconstructs the data packets from both kinds of packets. The LOOPS Egress currently resequences the forwarded and reconstructed packets, so they are

passed on in-order when the lost packets are recoverable within the source block.

The LOOPS Nodes need to agree on the use of FEC block mode and on the specific FEC Encoding ID to use; this is currently done by configuration.

#### Appendix B. Transparent mode

This appendix defines a very different way to provide the LOOPS services, "transparent mode". (We call the protocol described in the main body of the document "encapsulated mode".)

In transparent mode, the idea is that LOOPS does not meddle with the forward transmission of data packets, but runs on the side exchanging additional information.

An implementation could be based on conventional forwarding switches that just provide a copy of the ingress and egress packet stream to the LOOPS implementations. The LOOPS process would occasionally inject recovered packets back into the LOOPS egress node's forwarding switch, see Figure 3.

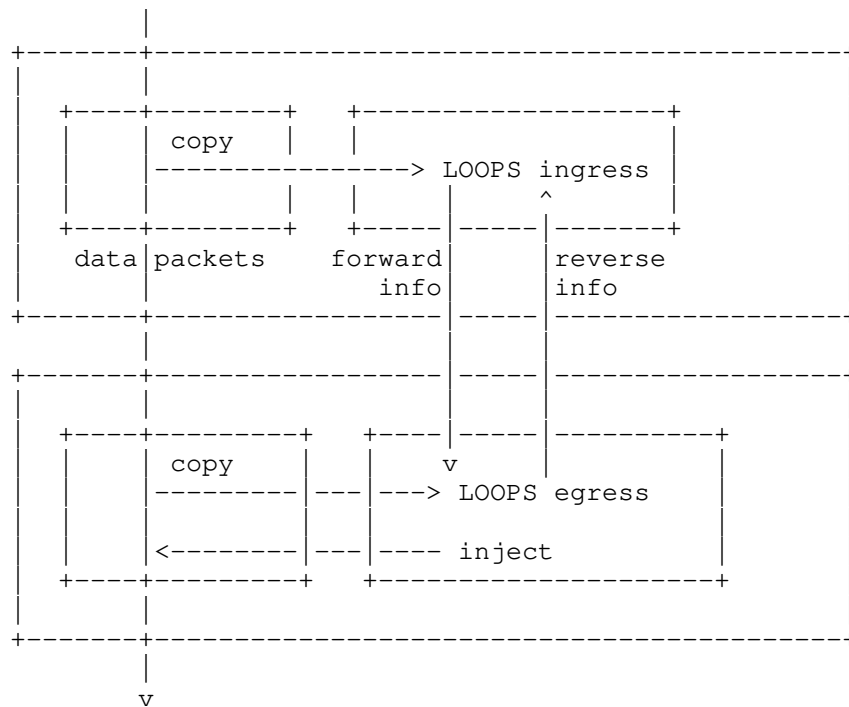


Figure 3: LOOPS Transparent Mode

The obvious advantage of transparent mode is that no encapsulation is needed, reducing processing requirements and keeping the MTU unchanged. The obvious disadvantage is that no forward information can be provided with each data packet, so a replacement needs to be found for the PSN (packet sequence number) employed in encapsulated mode. Any forward information beyond the data packets is sent in separate packets exchanged directly between the LOOPS nodes.

#### B.1. Packet identification

Retransmission mode and FEC mode differ in their needs for packet identification. For retransmission mode, a somewhat probabilistic accuracy of the packet identification is sufficient, for FEC mode, packet identification should not make mistakes (as these would lead to faultily reconstructed packets).

In Retransmission mode, misidentification of a packet could lead to measurement errors as well as missed retransmission opportunities. The latter will be fixed end-to-end. The tolerance for measurement errors would influence the degree of accuracy that is aimed for.

Packet identification can be based on a cryptographic hash of the packet, computed in LOOPS ingress and egress using the same algorithm (excluding fields that can change in transit, such as TTL/hop limit). The hash can directly be used as a packet number, or it can be sent in the forward information together with a packet sequence number, establishing a mapping.

For probabilistic packet identification, it is almost always sufficient to hash the first few (say, 64) bytes of the packet; all known transport protocols keep sufficient identifying information in that part (and, for encrypted protocols, the entropy will be sufficient). Any collisions of the hash could be used to disqualify the packet for measurement purposes, minimizing the measurement errors; this could allow rather short packet identifiers in retransmission mode.

For FEC mode, the packet identification together with the per-packet FEC information needs to be sent in the (separate) forward information, so that a systematic code can be reconstructed. For retransmission mode, there is no need to send any forward information for most packets, or a mapping from packet identifiers to packet sequence numbers could be sent in the forward information (probably in some aggregated form). The latter would allow keeping the acknowledgement form described in the main body (with aggregate acknowledgement); otherwise, packet identifiers need to be acknowledged. With this change, the LOOPS egress will send reverse information as in the encapsulating LOOPS protocol.

## B.2. Generic information and protocol operation

With the changes outlined above, transparent mode operates just as encapsulated mode. If packet sequence numbers are not used, there is no use for block2 reverse information; if they are used, a new block3 needs to be defined that provides the mapping from packet identifiers to packet sequence numbers in the forward information. To avoid MTU reduction, some mechanism will be needed to encapsulate the actual FEC information (additional packets) in the forward information.

## Acknowledgements

Sami Boutros helped with sketching the use of Geneve (Section 6.1), and Tom Herbert helped with sketching the use of GUE (Section 6.2).

## Authors' Addresses

Michael Welzl  
University of Oslo  
PO Box 1080 Blindern  
Oslo N-0316  
Norway

Phone: +47 22 85 24 20  
Email: [michawe@ifi.uio.no](mailto:michawe@ifi.uio.no)

Carsten Bormann (editor)  
Universitaet Bremen TZI  
Postfach 330440  
Bremen D-28359  
Germany

Phone: +49-421-218-63921  
Email: [cabo@tzi.org](mailto:cabo@tzi.org)