Network Management Rearch Group
Internet-Draft
Intended status: Informational
Expires: 4 May 2021

D. Bogdanovic
X. Liu
Volta Networks
L.M. Contreras
Telefonica

31 October 2020

Multilevel configuration
draft-bogdanovic-multilevel-configuration-00

Abstract

This document describes issues caused by residual configurations in network devices and how multi-level configuration could potentially offer a solution.

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 4 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. Definitions and Acronyms

TCAM: Ternary Content Addressable Memory

2. Introduction

As network operators experience traffic and customer growth, the network device configurations are getting larger. All the config information, both network operator and customers, on the device is multiplexed into single file and the configuration differentiation belonging to different owners becomes harder. This leads to the operators not knowing why certain parts of the config are in the file. Another issue contributing to config growth are debugging sessions. Network operator enters the device and starts editing configuration. After the debug session is finnished, it is not unusual for debug configuration entries to stay in the config file indefinitely.

In order to solve this problem, some operators created central database with all the network configuration files that act as systems of record. If anything is to persist on the device in the network, it has to be in the central database. Still, this solution has not remedided the problem.

Both, vendors and operators, contribute to the problem:

* Vendors by keeping the configuration file structures as currently designed;

* Operators by allowing human operator to directly edit config file on the device.

Until the above two issues are solved, the residual configuration problem will persist and continue to waste expensive data plane resources (TCAM).

This draft authors are motivated to propose a solution from both sides, operator and vendor. Our initial idea is to keep the persistent configuration at minimum on the device. All network service configurations are generated on demand and are ephemeral. This requires a change to the config file structure, creating multi-level file structure with dependencies between different levels.Besides the residual configuration problem, there are other use cases that multi-level configuration can be applied, that are listed in this document.

3. Use cases

3.1. Service assurance

Service assurance is one of the critical operational aspects of the communication networks. As [I-D.claise-opsawg-service-assurance-architecture] states, services rely on multiple sub-services on top of the same underlying network, then service affection on any of those sub-services can propagate impacts to many other services in the network. In this respect, the multi-level network configuration approach could help on identifying by design the correlation among services and atomic functions in the network, simplifying the operation and providing a uniform framework across networks.

3.2. Network migrations and mergings

Quite often service providers get involved in complex procedures of network mergings or migrations. Either driven by simplification of existing networks, introduction of new services, rationalization of multiple infrastructures, acquisition of other providers, etc., all of them imply both the introduction and removal of distinct configurations of multiple purposes. Apart of the complexity and difficulty of converging to a common and unique approach, these procedures could impact service continuity. In this sense, multi-level network configuration could highly simplify the process. First, by dividing the problem in smaller pieces, dealing with the issue per configuration level instead of considering the whole configuration. And second, by allowing incremental execution of the process by acting on particular levels each time.

3.3. Network slicing

Network slices are expected to provide tailored networks that can accommodate services with specific characteristics and service level objectives (SLOs) [I-D.nsdt-teas-ietf-network-slice-definition]. In this respect, the multi-level network configuration approach can be leveraged as a mean for deploying particular IETF network slices, facilitating the instantiation, operation and decommissioning of the slice in a straightforward manner.

3.4. Zero touch provisioning

[RFC8886] proposes a mechanism for remotely auto-installing configurations on network devices with proper confidentiality and security. Such mechanism is conceived for receiving initial configuration by the device, for a later completion of the configuration by other means. In this case, leveraging on multi-level network configuration could permit incremental deployment of configuration levels following a similar auto-installing approach, according to some configuration workflow as defined by the service provider.

4. Security Considerations

TBD

5. IANA Considerations

This document currently has no items for IANA considerations.

6. Acknowledgements

7. Change log [RFC Editor: Please remove]

8. Informative References

[I-D.claise-opsawg-service-assurance-architecture]
Claise, B., Quilbeuf, J., Fathi, Y., Lopez, D., and D. Voyer, "Service Assurance for Intent-based Networking Architecture", Work in Progress, Internet-Draft, draft-claise-opsawg-service-assurance-architecture-03, 27 July 2020, <http://www.ietf.org/internet-drafts/draft-claise-opsawg-service-assurance-architecture-03.txt>.

[I-D.nsdt-teas-ietf-network-slice-definition]
Rokui, R., Homma, S., Makhijani, K., Contreras, L., and J. Tantsura, "Definition of IETF Network Slices", Work in Progress, Internet-Draft, draft-nsdt-teas-ietf-network-slice-definition-00, 21 October 2020, <http://www.ietf.org/internet-drafts/draft-nsdt-teas-ietf-network-slice-definition-00.txt>.

[RFC8886] Kumari, W. and C. Doyle, "Secure Device Install", RFC 8886, DOI 10.17487/RFC8886, September 2020, <https://www.rfc-editor.org/info/rfc8886>.

Authors' Addresses

Dean Bogdanovic
Volta Networks

Email: dean@voltanet.io

Xufeng Liu
Volta Networks

Email: xufeng@voltant.io

Luis M. Contreras
Telefonica

Email: luismiguel.contrerasmurillo@telefonica.com

Network Working Group
Internet-Draft
Intended status: Experimental
Expires: April 18, 2021

H. Chen
China Telecom
Z. Hu
Huawei Technologies
H. Chen
Futurewei
X. Geng
Huawei Technologies

October 15, 2020

SRv6 Midpoint Protection
draft-chen-rtgwg-srv6-midpoint-protection-03

Abstract

The current local repair mechanism, e.g., TI-LFA, allows local repair actions on the direct neighbors of the failed node to temporarily route traffic to the destination. This mechanism could not work properly when the failure happens in the destination point or the link connected to the destination. In SRv6 TE, the IPv6 destination address in the outer IPv6 header could be the dedicated endpoint of the TE path rather than the destination of the TE path. When the endpoint fails, local repair couldn't work on the direct neighbor of the failed endpoint either. This document defines midpoint protection, which enables the direct neighbor of the failed endpoint to do the function of the endpoint, replace the IPv6 destination address to the other endpoint, and choose the next hop based on the new destination address.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any

time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on April 18, 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

The current mechanism, e.g., TI-LFA ([I-D.ietf-rtgwg-segment-routing-ti-lfa]), allows local repair actions on the direct neighbors of the failed node to temporarily route traffic to the destination. This mechanism could not work properly when the failure happens in the destination point or the link connected to the destination. In SRv6 TE, the IPv6 destination address in the outer IPv6 header could be the dedicated endpoint of the TE path rather than the destination of the TE path

([I-D.ietf-spring-srv6-network-programming]). When the endpoint fails, local repair couldn't work on the direct neighbor of the failed endpoint either. This document defines midpoint protection, which enables the direct neighbor of the failed endpoint to do the function of the endpoint, replace the IPv6 destination address to the other endpoint, and choose the next hop based on the new destination address.

2. SRv6 Midpoint Protection Mechanism

When an endpoint node fails, the packet needs to bypass the failed endpoint node and be forwarded to the next endpoint node of the failed endpoint. On the Repair Node (i.e., the previous hop of the failed endpoint node), it performs the proxy forwarding as follows :

- Outbound interface failure happens in the Repair Node;

Case 1: Route to the failed endpoint could be found in the FIB of Repair Node:

- If the Repair Node is not directly connected to the failed endpoint, the normal Ti-LFA is executed;

- If the Repair Node is directly connected to the failed endpoint, the Repair Node forwards the packets through a bypass to the failed endpoint, changing the IPv6 destination address with the IPv6 address of the next, the last or other reasonable endpoint nodes, which could avoid going throw the failed endpoint.

Case 2: Route to the failed endpoint could not be found in the FIB of Repair Node:

- Repair Node forwards the packets through a bypass of the failed endpoint to the next, the last or other reasonable endpoint node directly . There is no need to check whether the failed endpoint node is directly connected to the Repair Node or not.

3. SRv6 Midpoint Protection Example

The topology shown in Figure 1 illustrates an example of network topology with SRv6 enabled on each node.

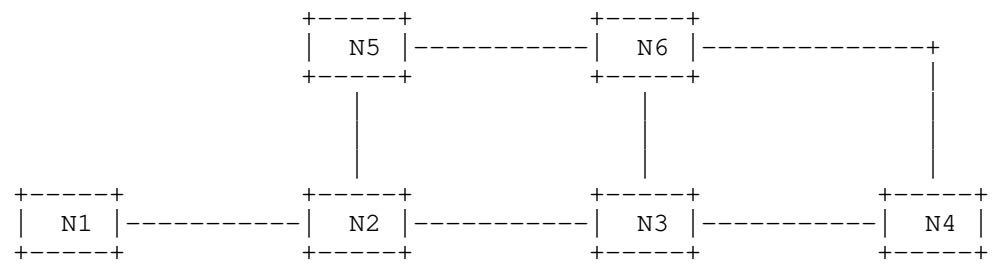


Figure 1: An example of midpoint protection

In this document, an end SID at node n with locator block B is represented as B:n. An end.x SID at node n towards node k with locator block B is represented as B:n:k. A SID list is represented as <S1, S2, S3> where S1 is the first SID to visit, S2 is the second SID to visit and S3 is the last SID to visit along the SRv6 TE path.

In the reference topology:

Node N1 is an ingress node of SRv6 domain. Node N1 steers a packet into a segment list < B:3, B:4>.

When Node N3 fails, the packet needs to bypass the failed endpoint node and be forwarded to the next endpoint node after the failed endpoint in the TE path. When outbound interface failure happens in the Repair Node (which is not limited to the previous hop node of the failed endpoint node), it performs the proxy forwarding as follows,:

For node N2, if the outbound interface to the endpoint B:3 is failed before IGP converges:

- Because node N2, as a Repair Node, is connected to the failed endpoint B:3 directly, node N2 forwards the packets through a bypass of the failed endpoint, changing the IPv6 destination address with the next sid B:4. N2 detects the failure of outbound interface to B:4 in the current route, it could use the normal Ti-LFA repait path to forward the packet, because it is not directly connected to the node N4. N2 encapsulates the packet with the segment list < B:5:6> as a repair path.

For node N1, route to the failed endpoint N3 could not be found in the FIB after IGP converges:

- Node N1, as a Repair Node, forwards the packets through a bypass of the failed endpoint to the next or endpoint node (e.g., N4) directly. There is no need to check whether the failed endpoint

node is directly connected to N1. N1 changes the IPv6 destination address with the next sid B:4. Since IGP has completed convergence, it forwards packets directly based on the IGP SPF path

4. SRv6 Midpoint Protection Behavior

4.1. Transit Node as Repair Node

When the Repair Node is a transit node, it provides fast protection against the endpoint node failure as follows after looking up the FIB.

```
IF the primary outbound interface used to forward the packet failed
  IF NH = SRH && SL != 0, and
     the failed endpoint is directly connected to the Repair Node THEN
    SL decreases*; update the IPv6 DA with SRH[SL];
    FIB lookup on the updated DA;
    forward the packet according to the matched entry;
  ELSE
    forward the packet according to the backup nexthop;
ELSE // there is no FIB entry for forwarding the packet
  IF NH = SRH && SL != 0 THEN
    SL decreases*; update the IPv6 DA with SRH[SL];
    FIB lookup on the updated DA;
    forward the packet according to the matched entry;
  ELSE
    drop the packet;
```

*: SL could decrease any dedicated value from [1-N], where N is the current value of SL.
The case is similar in the following examples.

4.2. Endpoint Node as Repair Node

When a node N receives a packet, if the destination address (DA) of the packet is a local END SID, then node N is an endpoint node. When the Repair Node is an endpoint node, it provides fast protections for the failure through executing the following procedure after looking up the FIB for the updated DA.

```
IF the primary outbound interface used to forward the packet failed
  IF NH = SRH && SL != 0, and
     the failed endpoint is directly connected to the Repair Node THEN
    SL decreases; update the IPv6 DA with SRH[SL];
    FIB lookup on the updated DA;
    forward the packet according to the matched entry;
  ELSE
    forward the packet according to the backup nexthop;
ELSE // there is no FIB entry for forwarding the packet
  IF NH = SRH && SL != 0 THEN
    SL decreases; update the IPv6 DA with SRH[SL];
    FIB lookup on the updated DA;
    forward the packet according to the matched entry;
  ELSE
    drop the packet;
ELSE
  forward accordingly to the matched entry;
```

4.3. Endpoint x Node as Repair Node

An endpoint node with cross-connect (End.X for short) is an endpoint node with an array of layer 3 adjacencies. When a node N receives a packet, if the destination address (DA) of the packet is a local END.X SID, then node N as Repair Node provides fast protections for the failure through executing the following procedure after updating DA.

```
IF the layer-3 adjacency interface is down THEN
  FIB lookup on the updated DA;
  IF the primary interface used to forward the packet failed THEN
    IF NH = SRH && SL != 0, and
       the failed endpoint directly connected to the Repair Node THEN
      SL decreases; update the IPv6 DA with SRH[SL];
      FIB lookup on the updated DA;
      forward the packet according to the matched entry;
    ELSE
      forward the packet according to the backup nexthop;
  ELSE // there is no FIB entry for forwarding the packet
    IF NH = SRH && SL != 0 THEN
      SL decreases; update the IPv6 DA with SRH[SL];
      FIB lookup on the updated DA;
      forward the packet according to the matched entry;
    ELSE
      drop the packet;
ELSE
  forward accordingly to the matched entry;
```

5. Determining whether the Endpoint could Be Bypassed

SRv6 Midpoint Protection provides a mechanism to bypass a failed endpoint. But in some scenarios, some important functions may be implemented in the bypassed failed endpoints that should not be bypassed, such as firewall functionality or In-situ Flow Information Telemetry of a specified path. Therefore, a mechanism is needed to indicate whether an endpoint can be bypassed or not. [I-D.li-rtgwg-enhanced-ti-lfa] provides method to determine whether enbale SRv6 midpoint protection or not by defining a "no bypass" flag for the SIDs in IGP.

6. Security Considerations

This section reviews security considerations related to SRv6 Midpoint protection processing discussed in this document.To ensure that the Repair node does not modify the SRH header Encapsulated by nodes outside the SRv6 Domain.Only the segment within the SRH is same domain as the repair node. So it is necessary to check the skipped segment have same block as repair node.

7. IANA Considerations

This document makes no request of IANA.

Note to RFC Editor: this section may be removed on publication as an RFC.

8. Acknowledgements

9. References

9.1. Normative References

[I-D.hu-spring-segment-routing-proxy-forwarding]
Hu, Z., Chen, H., Yao, J., Bowers, C., and Y. Zhu, "SR-TE Path Midpoint Protection", draft-hu-spring-segment-routing-proxy-forwarding-11 (work in progress), August 2020.

[I-D.ietf-isis-segment-routing-extensions]
Previdi, S., Ginsberg, L., Filsfils, C., Bashandy, A., Gredler, H., and B. Decraene, "IS-IS Extensions for Segment Routing", draft-ietf-isis-segment-routing-extensions-25 (work in progress), May 2019.

[I-D.ietf-lsr-isis-srv6-extensions]
Psenak, P., Filsfils, C., Bashandy, A., Decraene, B., and Z. Hu, "IS-IS Extension to Support Segment Routing over IPv6 Dataplane", draft-ietf-lsr-isis-srv6-extensions-11 (work in progress), October 2020.

[I-D.ietf-lsr-ospfv3-srv6-extensions]
Li, Z., Hu, Z., Cheng, D., Talaulikar, K., and P. Psenak, "OSPFv3 Extensions for SRv6", draft-ietf-lsr-ospfv3-srv6-extensions-01 (work in progress), August 2020.

[I-D.ietf-ospf-segment-routing-extensions]
Psenak, P., Previdi, S., Filsfils, C., Gredler, H., Shakir, R., Henderickx, W., and J. Tantsura, "OSPF Extensions for Segment Routing", draft-ietf-ospf-segment-routing-extensions-27 (work in progress), December 2018.

[I-D.ietf-spring-srv6-network-programming]
Filsfils, C., Camarillo, P., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "SRv6 Network Programming", draft-ietf-spring-srv6-network-programming-24 (work in progress), October 2020.

[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>.

[RFC7356] Ginsberg, L., Previdi, S., and Y. Yang, "IS-IS Flooding Scope Link State PDUs (LSPs)", RFC 7356, DOI 10.17487/RFC7356, September 2014, <https://www.rfc-editor.org/info/rfc7356>.

9.2. Informative References

[I-D.hegde-spring-node-protection-for-sr-te-paths]
Hegde, S., Bowers, C., Litkowski, S., Xu, X., and F. Xu, "Node Protection for SR-TE Paths", draft-hegde-spring-node-protection-for-sr-te-paths-07 (work in progress), July 2020.

[I-D.ietf-rtgwg-segment-routing-ti-lfa]
Litkowski, S., Bashandy, A., Filsfils, C., Decraene, B., Francois, P., Voyer, D., Clad, F., and P. Camarillo, "Topology Independent Fast Reroute using Segment Routing", draft-ietf-rtgwg-segment-routing-ti-lfa-04 (work in progress), August 2020.

[I-D.ietf-spring-segment-routing-policy]
Filsfils, C., Talaulikar, K., Voyer, D., Bogdanov, A., and P. Mattes, "Segment Routing Policy Architecture", draft-ietf-spring-segment-routing-policy-08 (work in progress), July 2020.

[I-D.li-rtgwg-enhanced-ti-lfa]
Li, C. and Z. Hu, "Enhanced Topology Independent Loop-free Alternate Fast Re-route", draft-li-rtgwg-enhanced-ti-lfa-02 (work in progress), August 2020.

[I-D.sivabalan-pce-binding-label-sid]
Sivabalan, S., Filsfils, C., Tantsura, J., Hardwick, J., Previdi, S., and C. Li, "Carrying Binding Label/Segment-ID in PCE-based Networks.", draft-sivabalan-pce-binding-label-sid-07 (work in progress), July 2019.

[RFC5462] Andersson, L. and R. Asati, "Multiprotocol Label Switching (MPLS) Label Stack Entry: "EXP" Field Renamed to "Traffic Class" Field", RFC 5462, DOI 10.17487/RFC5462, February 2009, <https://www.rfc-editor.org/info/rfc5462>.

Authors' Addresses

Huanan Chen
China Telecom
109, West Zhongshan Road, Tianhe District
Guangzhou 510000
China

Email: chenhuan6@chinatelecom.cn

Zhibo Hu
Huawei Technologies
Huawei Bld., No.156 Beiqing Rd.
Beijing 100095
China

Email: huzhibo@huawei.com

Huaimo Chen
Futurewei
Boston, MA
USA

Email: Huaimo.chen@futurewei.com

Xuesong Geng
Huawei Technologies

Email: gengxuesong@huawei.com

rtgwg
Internet-Draft
Intended status: Informational
Expires: May 3, 2021

L. Geng
P. Liu
China Mobile
P. Willis
BT

October 30, 2020

Dynamic-Anycast in Compute First Networking (CFN-Dyncast) Use Cases and Problem Statement
draft-geng-rtgwg-cfn-dyncast-ps-usecase-00

Abstract

Service providers are exploring the edge computing to achieve better response time, control over data and carbon energy saving by moving the computing services towards the edge of the network in scenarios of 5G MEC (Multi-access Edge Computing), virtualized central office, and others. Providing services by sharing computing resources from multiple edges is emerging and becoming more and more useful for computationally intensive tasks. The service nodes attached to multiple edges normally have two key features, service equivalency and service dynamism. Ideally they should serve the service in a computational balanced way. However lots of approaches dispatch the service in a static way, e.g., to the geographically closest edge, and they may cause unbalanced usage of computing resources at edges which further degrades user experience and system utilization. This draft provides an overview of scenarios and problems associated.

Networking taking account of computing resource metrics as one of its top parameters is called Compute First Networking (CFN) in this document. The document identifies several key areas which require more investigations in architecture and protocol to achieve the balanced computing and networking resource utilization among edges in CFN.

Status of This Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any

time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on May 3, 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

Edge computing aims to provide better response times and transfer rate, with respect to Cloud Computing, by moving the computing towards the edge of the network. Edge computing can be built on industrial PCs, embedded systems, gateways and others. They are put close to the end user. There is an emerging requirement that multiple edge sites (called edges too in this document) are deployed

at different locations to provide the service. There are millions of home gateways, thousands of base stations and hundreds of central offices in a city that can serve as candidate edges for hosting service nodes. Depending on the location of the edge and its capacity, each edge has different computing resources to be used for a service. At peak hour, computing resources attached to a client's closest edge site may not be sufficient to handle all the incoming service demands. Longer response time or even demand dropping can be experienced by the user. Increasing the computing resources hosted on each edge site to the potential maximum capacity is neither feasible nor economical.

Some user devices are purely battery-driven. Offloading the computation intensive processing to the edge can save the battery. Moreover the edge may use the data set (for the computation) that may not exist on the user device because of the size of data pool or data governance reasons.

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

DNS-based load balancing usually configures a domain in Domain Name System (DNS) such that client requests to the domain are distributed across a group of servers. It usually provides several IP addresses for a domain name. The traditional techniques to manage the overall load balancing process of clients issuing requests including choose-the-closest or round-robin. The are relatively static which may cause the unbalanced workload distribution in terms of network load and computational load.

There are some dynamic ways which tries to distribute the request to the server with the best available resources and minimal load. They usually require L4-L7 handling of the packet processing. It is not an efficient approach for huge number of short connections. At the same time, such approaches can hardly get network status in real time. Therefore the choice of the service node is almost entirely determined by the computing status, rather than the comprehensive consideration of both computing and network.

Networking taking account of computing resource metrics as one of its top parameters is called Compute First Networking (CFN) in this document. Edge site can interact with each other to provide network-based edge computing service dispatching to achieve better load balancing in CFN. Both computing load and network status are network visible resources.

A single service has multiple instances attached to multiple edge computing sites is conceptually like anycast in network language. Because of the dynamic and anycast aspects of the problem, jointly with the CFN deployment, we generally refer to it in this document as CFN-Dyncast, as for Compute First Networking Dynamic Anycast. This draft describes usage scenarios, problem space and key areas of CFN-Dyncast.

2. Definition of Terms

CFN: Compute First Networking. Networking architecture taking account of computing resource metrics as one of its top parameters to achieve flexible load management and performance optimizations in terms of both network and computing resources.

CFN-Dyncast: Compute First Networking Dynamic Anycast. The dynamic and anycast aspects of the architecture in a CFN deployment.

3. Main Use-Cases

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

3.1. Cloud Based Recognition in Augmented Reality (AR)

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

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

3.2. Connected Car

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

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

3.3. Cloud Virtual Reality (VR)

Cloud VR introduces the concept and technology of cloud computing and rendering into VR applications. Edge cloud helps encode/decode and rendering in this scenario. The end device usually only uploads the posture or control information to the edge and then VR contents are rendered in edge cloud. The video and audio outputs generated from edge cloud are encoded, compressed, and transmitted back to the end device or further transmitted to central data center via high bandwidth network.

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

Edge site may use CPU or GPU for encode/decode. GPU usually has better performance but CPU is more simple and straight forward for use. Edges have different computing resources in terms of CPU and GPU physically deployed. Available remaining resource determines if a gaming instance can be started. The instance CPU, GPU and memory utilization has a high impact on the processing delay on encoding, decoding and rendering. At the same time, the network path quality to the edge site is a key for user experience on quality of audio/video and game command response time.

Cloud VR service brings challenging requirements on both network and computing so that the edge node to serve a service demand has to be carefully selected to make sure it has sufficient computing resource and good network path.

4. Requirements

This document mainly targets at the typical edge computing scenarios with two key features, service equivalency and service dynamism.

- Service equivalency: A service is provided by one or more service instances, providing an equivalent service functionality to clients, while the existence of several instances is (possibly across multiple edges) is to ensure better scalability and availability

- Service dynamism: A single instance has very dynamic resources over time to serve a service demand. Its dynamism is affected by computing resource capability and load, network path quality, and etc. The balancing mechanisms should adapt to the service dynamism quickly and seamlessly. Failover kind of switching is not desired.

5. Problems Statement

A service demand should be routed to the most suitable edge and further to the service instance in real time among the multiple edges with service equivalency and dynamism. Existing mechanisms use one or more of the following ways and each of them has issues associated.

- Use the least network cost as metric to select the edge. Issue: Computing information is a key to be considered in edge computing, and it is not included here.

- Use geographical location deduced from IP prefix, pick the closest edge. Issue: Edges are not so far apart in edge computing scenario. Either hard to be deduced from IP address or the location is not the key distinguisher.

- Health check in an infrequent base (>1s) to reflect the service node status, and switch when fail-over. Issue: Health check is very different from computing status information of service instance. It is too coarse granularity.

- Application layer randomly picks or uses round-robin way to pick a service node. Issue: It may share the load across multiple service instances in terms of the computing capacity, the network cost variance is barely considered. Edges can be deployed in

different cities which are not equal cost paths to a client. Therefore network status is also a major concern.

- Global resolver and early binding (DNS-based load balancing): Client queries a global resolver or load balancer first and gets the exact server's address. And then steer traffic using that address as binding address. It is called early binding because an explicit binding address query has to be performed before sending user data. Issue: Firstly, it clashes with the service dynamism. Current resolver does not have the capability of such high frequent change of indirection to new instance based on the frequent change of each service instance. Secondly, edge computing flow can be short. One or two round trip would be completed. Out-of-band query for specific server address has high overhead as it takes one more round trips. As discussed in section 5.4 of [I-D.sarathchandra-coin-appcentres], the flexible re-routing to appropriate service instances out of a pool of available ones faces significant challenges when utilizing DNS for this purpose.

- Traditional anycast. Issue: Only works for single request/reply communication. No flow affinity guaranteed.

A network based dynamic anycast (Dyncast) architecture aims to address the following points in CFN (CFN-Dyncast).

5.1. Anycast based service addressing methodology

A unique service identifier is used by all the service instances for a specific service no matter which edge it attaches to. An anycast like addressing and routing methodology among multiple edges makes sure the data packet potentially can reach any of the edges with the service instance attached. At the same time, each service instance has its own unicast address to be used by the attaching edge to access the service. From service identifier (an anycast address) to a specific unicast address, the discovery and mapping methodology is required to allow in-band service instance and edge selection in real time in network.

5.2. Flow affinity

The traditional anycast is normally used for single request single response style communication as each packet is forwarded individually based on the forwarding table at the time. Packets may be sent to different places when the network status changes. CFN in edge computing requires multiple request multiple response style communication between the client and the service node. Therefore the

data plane must maintain flow affinity. All the packets from the same flow should go to the same service node.

5.3. Computing Aware Routing

Given that the current state of the art for routing is based on the network cost, computing resource and/or load information is not available or distributed at the network layer. At the same time, computing resource metrics are not well defined and understood by the network. They can be CPU/GPU capacity and load, number of sessions currently serving, latency of service process expected and the weights of each metric. Hence it is hard to make the best choice of the edge based on both computing and network metrics at the same time.

Computing information metric representation has to be agreed on by the participated edges and metrics are to be exchanged among them.

Network cost in current routing system does not change very frequently. However computing load is highly dynamic information. It changes rapidly with the number of sessions, CPU/GPU utilization and memory space. It has to be determined at what interval or event such information needs to be distributed among edges. More frequent distribution more accurate synchronization, but also more overhead.

Choosing the least path cost is the most common rule in routing. However, the logic does not work well in computing aware routing. Choosing the least computing load may result in oscillation. The least load edge can quickly be flooded by the huge number of new computing demands and soon become overloaded. Tidal effect may follow.

Depending on the usage scenario, computing information can be carried in BGP, IGP or SDN-like centralized way. More investigations in those solution spaces is to be elaborated in other documents. It is out of scope of this draft.

6. Summary

This document presents the CFN-Dyncast problem statement. CFN-Dyncast aims at leveraging the resources mobile providers have available at the edge of their networks. However, CFN-Dyncast aims at taking into account as well the dynamic nature of service demands and the availability of network resources so as to satisfy service requirements and load balance among service instances.

This also document illustrate some use cases problems and list the requirements for CFN-Dyncast. CFN-Dyncast architecture should

addresses how to distribute the computing resource information at the network layer and how to assure flow affinity in an anycast based service addressing environment.

7. Security Considerations

TBD

8. IANA Considerations

No IANA action is required so far.

9. Informative References

[I-D.sarathchandra-coin-appcentres]
Trossen, D., Sarathchandra, C., and M. Boniface, "In-Network Computing for App-Centric Micro-Services", draft-sarathchandra-coin-appcentres-03 (work in progress), October 2020.

Acknowledgements

The author would like to thank Yizhou Li, Luigi IANNONE and Dirk Trossen for their valuable suggestions to this document.

Authors' Addresses

Liang Geng
China Mobile

Email: gengliang@chinamobile.com

Peng Liu
China Mobile

Email: liupengyjy@chinamobile.com

Peter Willis
BT

Email: peter.j.willis@bt.com

Routing area
Internet-Draft
Intended status: Standards Track
Expires: May 19, 2021

S. Hegde
W. Lin
Juniper Networks Inc.
P. Shaofu
ZTE Corporation

November 15, 2020

Egress Protection for Segment Routing (SR) networks
draft-hegde-rtgwg-egress-protection-sr-networks-01

Abstract

This document specifies a Fast Reroute(FRR) mechanism for protecting IP/MPLS services that use Segment Routing (SR) paths for transport against egress node and egress link failures. The mechanism is based on egress protection framework described in [RFC8679]. The egress protection mechanism can be further simplified in Segment Routing networks with anycast SIDs and anycast Locators. This document addresses all kinds of networks that use Segment Routing transport such as SR-MPLS over IPv4, SR-MPLS over IPv6, SRv6 and SRm6.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

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

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

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

This Internet-Draft will expire on May 19, 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

Segment Routing Architecture as defined in [RFC8402] provides a simple and scalable MPLS control plane that removes state from transit nodes in the network. SRm6 as defined in [I-D.bonica-spring-sr-mapped-six] and SRv6 as defined in [I-D.ietf-spring-srv6-network-programming] provide Segment Routing transport in pure IPv6 networks where MPLS data plane is not used. End-to-End resiliency is very important to satisfy Service Level Agreements (SLA) such as 50ms convergence. The transport resiliency and fast rerouting are described in[I-D.ietf-rtgwg-segment-routing-ti-lfa] and [I-D.ietf-spring-segment-protection-sr-te-paths]. Egress node and egress link failures are not covered by these protection mechanisms. Egress node and link failures need to address moving the services to other nodes where the customer services are multi-homed. In traditional MPLS networks service labels (ex: L3VPN) are assigned

dynamically. The protector nodes need to learn the service labels advertised by primary nodes and build a local context table corresponding to each primary node that a node protects. This requires building local context tables and also specialized context table lookups as described in [RFC8679].

Egress protection can be simplified by statically assigning service labels on egress nodes. When a service is multi-homed to two or more egress nodes, the same service label can be assigned to the service on each egress node. This mechanism, coupled with using anycast SIDs for loopbacks, greatly simplifies the egress protection. Following the principles described in [RFC8679], this document specifies procedures which can be used to greatly simplify the operation of egress protection in a segment routing network. Egress protection for Multicast services is for FFS.

2. Egress Node Protection

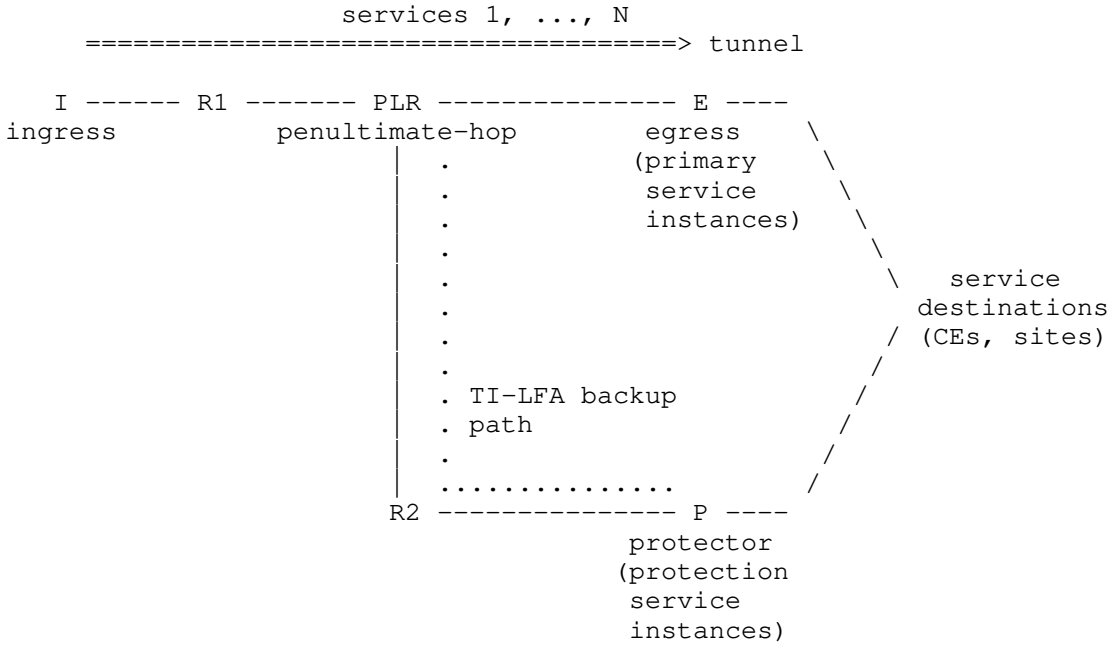


Figure 1: Reference topology

The reference topology from [RFC8679] has been reproduced in Figure 1 for ease of reading. The current document also uses the terminology defined in [RFC8679]. In the topology in Figure 1, service destinations are attached to two egress nodes. The two egress nodes

could be used in primary/protection mode or they could also be used in ECMP mode. When one of egress nodes fails, traffic should be switched to the other egress node and the convergence should be on the order of 50ms. The transport network is based on Segment Routing technology and could be using any of the SR-MPLS over IPv4, SR-MPLS over IPV6 , SRm6 or SRv6. The sections below describe the solution for each of the transport mechanisms.

2.1. SR-MPLS Networks

[RFC8402] describes the concept of anycast SIDs. Applying anycast SIDs to egress protection, the same IP address is configured as a loopback address on multiple egress nodes, and the same SID is advertised for this IP address. In the reference topology in Figure 1, E and P are associated with anycast loopback and corresponding anycast SID. The egress protected tunnel is considered logically destined to this anycast address and the egress protected tunnel always carries this anycast SID corresponding to the destination anycast address as the last SID. The egress protected service is hosted on both E and P. The egress protected tunnel can be used in primary/protector mode in which case, the anycast loopback MUST be advertised with better metric and the protector MUST advertise with an inferior metric. An egress protected service MUST advertise same service label from both E and P. The service label is assigned from the SRLB as defined in [RFC8402]. The egress node pairs that serve egress protected service MUST be able to allocate the same service label and hence MUST have overlapping local label space (SRLB) reserved for static assignment.

The TI-LFA procedures described in [I-D.ietf-rtgwg-segment-routing-ti-lfa] apply to the anycast prefixes. Based on the transport IGP topology, TI-LFA backup path is computed and programmed into the forwarding plane on the PLR nodes. The PLR SHOULD be configured to provide node protection for the failure. On the egress node E's failure, traffic on the PLR SHOULD be switched to the other egress node which is P. As the service label carried in the packet is understood by P as well, P will correctly send the traffic to the service destination.

Note that the micro-loop avoidance procedures as described in [I-D.bashandy-rtgwg-segment-routing-uloop] are applicable to anycast prefixes as well. When the anycast prefix is impacted by the failure event, a micro-loop avoiding path for the anycast prefix/anycast-SID will be programed during convergence. This mechanims does not affect the egress protection procedures described in this document.

The above procedure is applicable to SR-MPLS over IPv4 as well as SR-MPLS over IPv6 networks. In case of IPv4 networks, the anycast SIDs

are assigned to IPv4 loopbacks and in case of IPv6 networks, the anycast SIDs are assigned to IPv6 loopback addresses. The egress protected IP/MPLS services advertise the service prefix with the anycast address information in the message. In case of BGP based services such as L3vpn [RFC2547], the nexthop attribute carries the anycast address which is the logical tunnel destination address. On the ingress, when the service prefix is received, the service is mapped to the corresponding egress protected tunnel.

If some services are multi-homed to a different node for example, in the reference topology, if a service is multi-homed to E and another node P', then there SHOULD be another anycast address representing {E,P'}. The number of anycast loopbacks on a given node will be equal to the number of such {primary, protector} pairs a node belongs to. The egress protected service prefixes MUST carry the anycast address corresponding to the {primary, protector} pair in their next hop attribute.

When there is a single homed CE connected to the egress node, it SHOULD use a node loopback in the next hop attribute and should not use anycast loopback address.

2.2. SRm6 Networks

[I-D.bonica-spring-sr-mapped-six] defines segment routing applied to IPv6 networks which is optimized for high data rate forwarding. SRm6 control plane is very similar to SR-MPLS but it uses the IPv6 data plane. The egress node protection procedures described for SR-MPLS are applicable to SRm6 as well. Anycast loopback addresses are advertised and corresponding anycast SIDs are associated with the anycast addresses. The anycast SIDs in case of SRm6 are globally unique indices of size 16 or 32 bits.

The VPN services that require a label to identify the service are advertised as described in [I-D.ssangli-idr-bgp-vpn-srv6-plus]. The same PPSI value MUST be allocated to the service prefix on both the egress nodes on which the service is multi-homed. The TI-LFA procedures explained in Section 2.1 are applicable to SRm6 as well. After the CRH header is removed at the egress node, lookup is done based on PPSI which points to the correct service instance. Since the same PPSI is assigned on both nodes, the context table as described in [RFC8679] is not required to be built.

2.3. SRv6 Networks

[I-D.ietf-spring-srv6-network-programming] describes various types of SIDs used in SRv6 networks. The routing in the transport is based on the locators. Locators are most significant bits of the SID. In

order to achieve the egress protection functionality, anycast locators MUST be assigned on the egress nodes {E,P} where the service are multi-homed. The service SIDs MUST be derived from the anycast SID. The multi-homed service MUST be assigned with same service SID on both the egress nodes. It is recommended to provide mechanisms to statically configure the service SIDs which can easily serve the purpose of synchronized SID allocation on both nodes. As explained in the Section 2.1, if an egress node has services which are multi-homed to different nodes, then each such pair of node will need a separate locator assigned.

When there is a single homed CE connected to an egress, it MUST use a node specific locator to advertise service SID. It should not use service SIDs based on anycast locator

The TI-LFA procedure is applicable to anycast locators and each transport node in the transport IGP, installs a primary and backup path to the anycast locator. However, only the directly connected upstream PLR of the primary egress node will respond to the failure of the primary egress node and switch to the TI-LFA backup path. It is assumed that PLR has a backup path to alternate egress node which does not go via the primary egress node.

3. Egress Link Protection

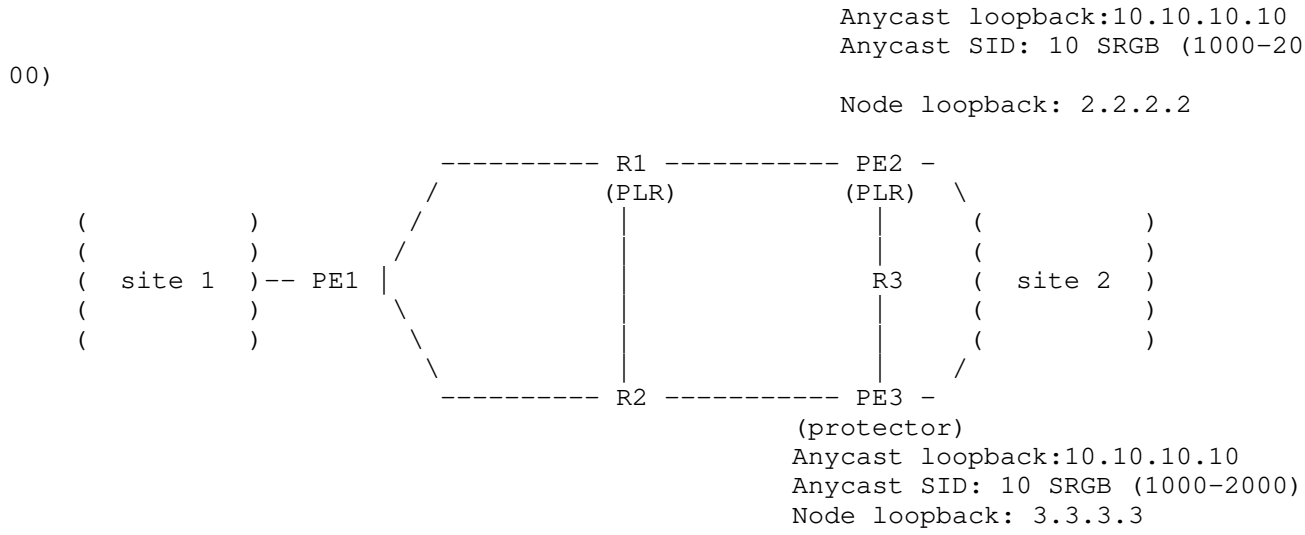


Figure 2: Egress Link Protection

The link from egress node towards the CE (service destination) fails, that failure needs to be protected. Egress link protection can be

achieved using similar means as egress node protection. section 6.2.2 of [I-D.ietf-rtgwg-bgp-pic] describes the procedures for protecting egress link failures in detail. When anycast ip address is used as BGP protocol nexthop, some additional considerations are required along with the procedures described in [I-D.ietf-rtgwg-bgp-pic]. In egress link failure case, egress node is the PLR and it has learned the service prefix from the other egress node. PLR egress node pre-establishes backup path to the other egress node and programs the forwarding plane with backup path. As the BGP based service prefixes advertise the anycast loopback address in the next hop attribute, the egress nodes will ignore the advertisement from other egress node. For Example, in the above Figure 2, when PE3 advertises a service prefix for site 2 with a next hop attribute of anycast loopback address, PE2 does not consider this advertisement and program a backup path towards PE3. To solve this problem, The egress nodes could advertise service prefixes with NEXT_HOP [RFC4271]attribute carrying anycast loopback as well as node specific loopback with a different RD [RFC2547].

4. Security Considerations

This document does not introduce any new security risks. For deploying this solution, security considerations described in [RFC8402], [I-D.bonica-spring-sr-mapped-six], [I-D.ietf-spring-srv6-network-programming] and [RFC8679] are applicable.

5. IANA Considerations

This document does not introduce any new IANA requests.

6. Acknowledgments

Thanks to Krzysztof Szarkowicz,Louis Chan and Chris Bowers for careful review and inputs.

7. References

7.1. Normative References

[I-D.bonica-spring-sr-mapped-six]
Bonica, R., Hegde, S., Kamite, Y., Alston, A., Henriques, D., Jalil, L., Halpern, J., Linkova, J., and G. Chen, "Segment Routing Mapped To IPv6 (SRm6)", draft-bonica-spring-sr-mapped-six-02 (work in progress), October 2020.

[I-D.ietf-spring-srv6-network-programming]
Filsfils, C., Camarillo, P., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "SRv6 Network Programming", draft-ietf-spring-srv6-network-programming-24 (work in progress), October 2020.

[RFC8402] Filsfils, 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>.

[RFC8679] Shen, Y., Jeganathan, M., Decraene, B., Gredler, H., Michel, C., and H. Chen, "MPLS Egress Protection Framework", RFC 8679, DOI 10.17487/RFC8679, December 2019, <https://www.rfc-editor.org/info/rfc8679>.

7.2. Informative References

[I-D.bashandy-rtgwg-segment-routing-uloop]
Bashandy, A., Filsfils, C., Litkowski, S., Decraene, B., Francois, P., and P. Psenak, "Loop avoidance using Segment Routing", draft-bashandy-rtgwg-segment-routing-uloop-09 (work in progress), June 2020.

[I-D.ietf-rtgwg-bgp-pic]
Bashandy, A., Filsfils, C., and P. Mohapatra, "BGP Prefix Independent Convergence", draft-ietf-rtgwg-bgp-pic-12 (work in progress), August 2020.

[I-D.ietf-rtgwg-segment-routing-ti-lfa]
Litkowski, S., Bashandy, A., Filsfils, C., Decraene, B., Francois, P., Voyer, D., Clad, F., and P. Camarillo, "Topology Independent Fast Reroute using Segment Routing", draft-ietf-rtgwg-segment-routing-ti-lfa-04 (work in progress), August 2020.

[I-D.ietf-spring-segment-protection-sr-te-paths]
Hegde, S., Bowers, C., Litkowski, S., Xu, X., and F. Xu, "Segment Protection for SR-TE Paths", draft-ietf-spring-segment-protection-sr-te-paths-00 (work in progress), September 2020.

[I-D.ssangli-idr-bgp-vpn-srv6-plus]
Ramachandra, S. and R. Bonica, "BGP based Virtual Private Network (VPN) Services over SRv6+ enabled IPv6 networks", draft-ssangli-idr-bgp-vpn-srv6-plus-02 (work in progress), July 2019.

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

[RFC2547] Rosen, E. and Y. Rekhter, "BGP/MPLS VPNs", RFC 2547, DOI 10.17487/RFC2547, March 1999, <https://www.rfc-editor.org/info/rfc2547>.

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

[RFC7911] Walton, D., Retana, A., Chen, E., and J. Scudder, "Advertisement of Multiple Paths in BGP", RFC 7911, DOI 10.17487/RFC7911, July 2016, <https://www.rfc-editor.org/info/rfc7911>.

Authors' Addresses

Shraddha Hegde
Juniper Networks Inc.
Exora Business Park
Bangalore, KA 560103
India

Email: shraddha@juniper.net

Wen Lin
Juniper Networks Inc.

Email: wlin@juniper.net

Peng Shaofu
ZTE Corporation

Email: peng.shaofu@zte.com.cn

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 8, 2018

C. Bormann
Universitaet Bremen TZI
B. Carpenter, Ed.
Univ. of Auckland
B. Liu, Ed.
Huawei Technologies Co., Ltd

July 7, 2017

A Generic Autonomic Signaling Protocol (GRASP)
draft-ietf-anima-grasp-15

Abstract

This document specifies the GeneRic Autonomic Signaling Protocol (GRASP), which enables autonomic nodes and autonomic service agents to dynamically discover peers, to synchronize state with each other, and to negotiate parameter settings with each other. GRASP depends on an external security environment that is described elsewhere. The technical objectives and parameters for specific application scenarios are to be described in separate documents. Appendices briefly discuss requirements for the protocol and existing protocols with comparable features.

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 January 8, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

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

Table of Contents

1. Introduction

The success of the Internet has made IP-based networks bigger and more complicated. Large-scale ISP and enterprise networks have become more and more problematic for human based management. Also, operational costs are growing quickly. Consequently, there are increased requirements for autonomic behavior in the networks. General aspects of autonomic networks are discussed in [RFC7575] and [RFC7576].

One approach is to largely decentralize the logic of network management by migrating it into network elements. A reference model for autonomic networking on this basis is given in [I-D.ietf-anima-reference-model]. The reader should consult this document to understand how various autonomic components fit together. In order to fulfill autonomy, devices that embody Autonomic Service

Agents (ASAs, [RFC7575]) have specific signaling requirements. In particular they need to discover each other, to synchronize state with each other, and to negotiate parameters and resources directly with each other. There is no limitation on the types of parameters and resources concerned, which can include very basic information needed for addressing and routing, as well as anything else that might be configured in a conventional non-autonomic network. The atomic unit of discovery, synchronization or negotiation is referred to as a technical objective, i.e, a configurable parameter or set of parameters (defined more precisely in Section 2.1).

Negotiation is an iterative process, requiring multiple message exchanges forming a closed loop between the negotiating entities. In fact, these entities are ASAs, normally but not necessarily in different network devices. State synchronization, when needed, can be regarded as a special case of negotiation, without iteration. Both negotiation and synchronization must logically follow discovery. More details of the requirements are found in Appendix E.
Section 2.3 describes a behavior model for a protocol intended to support discovery, synchronization and negotiation. The design of GeneRic Autonomic Signaling Protocol (GRASP) in Section 2 of this document is based on this behavior model. The relevant capabilities of various existing protocols are reviewed in Appendix F.

The proposed discovery mechanism is oriented towards synchronization and negotiation objectives. It is based on a neighbor discovery process on the local link, but also supports diversion to peers on other links. There is no assumption of any particular form of network topology. When a device starts up with no pre-configuration, it has no knowledge of the topology. The protocol itself is capable of being used in a small and/or flat network structure such as a small office or home network as well as in a large professionally managed network. Therefore, the discovery mechanism needs to be able to allow a device to bootstrap itself without making any prior assumptions about network structure.

Because GRASP can be used as part of a decision process among distributed devices or between networks, it must run in a secure and strongly authenticated environment.

In realistic deployments, not all devices will support GRASP. Therefore, some autonomic service agents will directly manage a group of non-autonomic nodes, and other non-autonomic nodes will be managed traditionally. Such mixed scenarios are not discussed in this specification.

2. GRASP Protocol Overview

2.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119] when they appear in ALL CAPS. When these words are not in ALL CAPS (such as "should" or "Should"), they have their usual English meanings, and are not to be interpreted as [RFC2119] key words.

This document uses terminology defined in [RFC7575].

The following additional terms are used throughout this document:

- Discovery: a process by which an ASA discovers peers according to a specific discovery objective. The discovery results may be different according to the different discovery objectives. The discovered peers may later be used as negotiation counterparts or as sources of synchronization data.

- Negotiation: a process by which two ASAs interact iteratively to agree on parameter settings that best satisfy the objectives of both ASAs.

- State Synchronization: a process by which ASAs interact to receive the current state of parameter values stored in other ASAs. This is a special case of negotiation in which information is sent but the ASAs do not request their peers to change parameter settings. All other definitions apply to both negotiation and synchronization.

- Technical Objective (usually abbreviated as Objective): A technical objective is a data structure, whose main contents are a name and a value. The value consists of a single configurable parameter or a set of parameters of some kind. The exact format of an objective is defined in Section 2.10.1. An objective occurs in three contexts: Discovery, Negotiation and Synchronization. Normally, a given objective will not occur in negotiation and synchronization contexts simultaneously.

  * One ASA may support multiple independent objectives.

  * The parameter(s) in the value of a given objective apply to a specific service or function or action. They may in principle be anything that can be set to a specific logical, numerical or string value, or a more complex data structure, by a network

node. Each node is expected to contain one or more ASAs which may each manage subsidiary non-autonomic nodes.

* Discovery Objective: an objective in the process of discovery. Its value may be undefined.

* Synchronization Objective: an objective whose specific technical content needs to be synchronized among two or more ASAs. Thus, each ASA will maintain its own copy of the objective.

* Negotiation Objective: an objective whose specific technical content needs to be decided in coordination with another ASA. Again, each ASA will maintain its own copy of the objective.

A detailed discussion of objectives, including their format, is found in Section 2.10.

- Discovery Initiator: an ASA that starts discovery by sending a discovery message referring to a specific discovery objective.

- Discovery Responder: a peer that either contains an ASA supporting the discovery objective indicated by the discovery initiator, or caches the locator(s) of the ASA(s) supporting the objective. It sends a Discovery Response, as described later.

- Synchronization Initiator: an ASA that starts synchronization by sending a request message referring to a specific synchronization objective.

- Synchronization Responder: a peer ASA which responds with the value of a synchronization objective.

- Negotiation Initiator: an ASA that starts negotiation by sending a request message referring to a specific negotiation objective.

- Negotiation Counterpart: a peer with which the Negotiation Initiator negotiates a specific negotiation objective.

- GRASP Instance: This refers to an instantiation of a GRASP protocol engine, likely including multiple threads or processes as well as dynamic data structures such as a discovery cache, running in a given security environment on a single device.

- GRASP Core: This refers to the code and shared data structures of a GRASP instance, which will communicate with individual ASAs via a suitable Application Programming Interface (API).

- Interface or GRASP Interface: Unless otherwise stated, these refer to a network interface – which might be physical or virtual – that a specific instance of GRASP is currently using. A device might have other interfaces that are not used by GRASP and which are outside the scope of the autonomic network.

2.2. High Level Deployment Model

A GRASP implementation will be part of the Autonomic Networking Infrastructure (ANI) in an autonomic node, which must also provide an appropriate security environment. In accordance with [I-D.ietf-anima-reference-model], this SHOULD be the Autonomic Control Plane (ACP) [I-D.ietf-anima-autonomic-control-plane]. As a result, all autonomic nodes in the ACP are able to trust each other. It is expected that GRASP will access the ACP by using a typical socket programming interface and the ACP will make available only network interfaces within the autonomic network. If there is no ACP, the considerations described in Section 2.5.1 apply.

There will also be one or more Autonomic Service Agents (ASAs). In the minimal case of a single-purpose device, these components might be fully integrated with GRASP and the ACP. A more common model is expected to be a multi-purpose device capable of containing several ASAs, such as a router or large switch. In this case it is expected that the ACP, GRASP and the ASAs will be implemented as separate processes, which are able to support asynchronous and simultaneous operations, for example by multi-threading.

In some scenarios, a limited negotiation model might be deployed based on a limited trust relationship such as that between two administrative domains. ASAs might then exchange limited information and negotiate some particular configurations.

GRASP is explicitly designed to operate within a single addressing realm. Its discovery and flooding mechanisms do not support autonomic operations that cross any form of address translator or upper layer proxy.

A suitable Application Programming Interface (API) will be needed between GRASP and the ASAs. In some implementations, ASAs would run in user space with a GRASP library providing the API, and this library would in turn communicate via system calls with core GRASP functions. Details of the API are out of scope for the present document. For further details of possible deployment models, see [I-D.ietf-anima-reference-model].

An instance of GRASP must be aware of the network interfaces it will use, and of the appropriate global-scope and link-local addresses.

In the presence of the ACP, such information will be available from the adjacency table discussed in [I-D.ietf-anima-reference-model]. In other cases, GRASP must determine such information for itself. Details depend on the device and operating system. In the rest of this document, the terms 'interfaces' or 'GRASP interfaces' refers only to the set of network interfaces that a specific instance of GRASP is currently using.

Because GRASP needs to work with very high reliability, especially during bootstrapping and during fault conditions, it is essential that every implementation continues to operate in adverse conditions. For example, discovery failures, or any kind of socket exception at any time, must not cause irrecoverable failures in GRASP itself, and must return suitable error codes through the API so that ASAs can also recover.

GRASP must not depend upon non-volatile data storage. All run time error conditions, and events such as address renumbering, network interface failures, and CPU sleep/wake cycles, must be handled in such a way that GRASP will still operate correctly and securely (Section 2.5.1) afterwards.

An autonomic node will normally run a single instance of GRASP, used by multiple ASAs. Possible exceptions are mentioned below.

2.3. High Level Design

This section describes the behavior model and general design of GRASP, supporting discovery, synchronization and negotiation, to act as a platform for different technical objectives.

- A generic platform:

  The protocol design is generic and independent of the synchronization or negotiation contents. The technical contents will vary according to the various technical objectives and the different pairs of counterparts.

- Normally, a single main instance of the GRASP protocol engine will exist in an autonomic node, and each ASA will run as an independent asynchronous process. However, scenarios where multiple instances of GRASP run in a single node, perhaps with different security properties, are possible (Section 2.5.2). In this case, each instance MUST listen independently for GRASP link-local multicasts, and all instances MUST be woken by each such multicast, in order for discovery and flooding to work correctly.

- Security infrastructure:

  As noted above, the protocol itself has no built-in security functionality, and relies on a separate secure infrastructure.

- Discovery, synchronization and negotiation are designed together:

  The discovery method and the synchronization and negotiation methods are designed in the same way and can be combined when this is useful, allowing a rapid mode of operation described in Section 2.5.4. These processes can also be performed independently when appropriate.

  * Thus, for some objectives, especially those concerned with application layer services, another discovery mechanism such as the future DNS Service Discovery [RFC7558] MAY be used. The choice is left to the designers of individual ASAs.

- A uniform pattern for technical objectives:

  The synchronization and negotiation objectives are defined according to a uniform pattern. The values that they contain could be carried either in a simple binary format or in a complex object format. The basic protocol design uses the Concise Binary Object Representation (CBOR) [RFC7049], which is readily extensible for unknown future requirements.

- A flexible model for synchronization:

  GRASP supports synchronization between two nodes, which could be used repeatedly to perform synchronization among a small number of nodes. It also supports an unsolicited flooding mode when large groups of nodes, possibly including all autonomic nodes, need data for the same technical objective.

  * There may be some network parameters for which a more traditional flooding mechanism such as DNCP [RFC7787] is considered more appropriate. GRASP can coexist with DNCP.

- A simple initiator/responder model for negotiation:

  Multi-party negotiations are very complicated to model and cannot readily be guaranteed to converge. GRASP uses a simple bilateral model and can support multi-party negotiations by indirect steps.

- Organizing of synchronization or negotiation content:

  The technical content transmitted by GRASP will be organized according to the relevant function or service. The objectives for different functions or services are kept separate, because they may be negotiated or synchronized with different counterparts or have different response times. Thus a normal arrangement would be a single ASA managing a small set of closely related objectives, with a version of that ASA in each relevant autonomic node. Further discussion of this aspect is out of scope for the current document.

- Requests and responses in negotiation procedures:

  The initiator can negotiate a specific negotiation objective with relevant counterpart ASAs. It can request relevant information from a counterpart so that it can coordinate its local configuration. It can request the counterpart to make a matching configuration. It can request simulation or forecast results by sending some dry run conditions.

  Beyond the traditional yes/no answer, the responder can reply with a suggested alternative value for the objective concerned. This would start a bi-directional negotiation ending in a compromise between the two ASAs.

- Convergence of negotiation procedures:

  To enable convergence, when a responder suggests a new value or condition in a negotiation step reply, it should be as close as possible to the original request or previous suggestion. The suggested value of later negotiation steps should be chosen between the suggested values from the previous two steps. GRASP provides mechanisms to guarantee convergence (or failure) in a small number of steps, namely a timeout and a maximum number of iterations.

- Extensibility:

  GRASP intentionally does not have a version number, and can be extended by adding new message types and options. The Invalid Message (M_INVALID) will be used to signal that an implementation does not recognize a message or option sent by another

implementation. In normal use, new semantics will be added by defining new synchronization or negotiation objectives.

2.4. Quick Operating Overview

An instance of GRASP is expected to run as a separate core module, providing an API (such as [I-D.liu-anima-grasp-api]) to interface to various ASAs. These ASAs may operate without special privilege, unless they need it for other reasons (such as configuring IP addresses or manipulating routing tables).

The GRASP mechanisms used by the ASA are built around GRASP objectives defined as data structures containing administrative information such as the objective's unique name, and its current value. The format and size of the value is not restricted by the protocol, except that it must be possible to serialize it for transmission in CBOR, which is no restriction at all in practice.

GRASP provides the following mechanisms:

- A discovery mechanism (M_DISCOVERY, M_RESPONSE), by which an ASA can discover other ASAs supporting a given objective.

- A negotiation request mechanism (M_REQ_NEG), by which an ASA can start negotiation of an objective with a counterpart ASA. Once a negotiation has started, the process is symmetrical, and there is a negotiation step message (M_NEGOTIATE) for each ASA to use in turn. Two other functions support negotiating steps (M_WAIT, M_END).

- A synchronization mechanism (M_REQ_SYN), by which an ASA can request the current value of an objective from a counterpart ASA. With this, there is a corresponding response function (M_SYNCH) for an ASA that wishes to respond to synchronization requests.

- A flood mechanism (M_FLOOD), by which an ASA can cause the current value of an objective to be flooded throughout the autonomic network so that any ASA can receive it. One application of this is to act as an announcement, avoiding the need for discovery of a widely applicable objective.

Some example messages and simple message flows are provided in Appendix D.

2.5. GRASP Protocol Basic Properties and Mechanisms

2.5.1. Required External Security Mechanism

GRASP does not specify transport security because it is meant to be adapted to different environments. Every solution adopting GRASP MUST specify a security and transport substrate used by GRASP in that solution.

The substrate MUST enforce sending and receiving GRASP messages only between members of a mutually trusted group running GRASP. Each group member is an instance of GRASP. The group members are nodes of a connected graph. The group and graph is created by the security and transport substrate and called the GRASP domain. The substrate must support unicast messages between any group members and (link-local) multicast messages between adjacent group members. It must deny messages between group members and non group members. With this model, security is provided by enforcing group membership, but any member of the trusted group can attack the entire network until revoked.

Substrates MUST use cryptographic member authentication and message integrity for GRASP messages. This can be end-to-end or hop-by-hop across the domain. The security and transport substrate MUST provide mechanisms to remove untrusted members from the group.

If the substrate does not mandate and enforce GRASP message encryption then any service using GRASP in such a solution MUST provide protection and encryption for message elements whose exposure could constitute an attack vector.

The security and transport substrate for GRASP in the ANI is the ACP. Unless otherwise noted, we assume this security and transport substrate in the remainder of this document. The ACP does mandate the use of encryption; therefore GRASP in the ANI can rely on GRASP message being encrypted. The GRASP domain is the ACP: all nodes in an autonomic domain connected by encrypted virtual links formed by the ACP. The ACP uses hop-by-hop security (authentication/ encryption) of messages. Removal of nodes relies on standard PKI certificate revocation or expiry of sufficiently short lived certificates. Refer to [I-D.ietf-anima-autonomic-control-plane] for more details.

As mentioned in Section 2.3, some GRASP operations might be performed across an administrative domain boundary by mutual agreement, without the benefit of an ACP. Such operations MUST be confined to a separate instance of GRASP with its own copy of all GRASP data structures running across a separate GRASP domain with a security and

transport substrate. In the most simple case, each point-to-point interdomain GRASP peering could be a separate domain and the security and transport substrate could be built using transport or network layer security protocols. This is subject to future specifications.

An exception to the requirements for the security and transport substrate exists for highly constrained subsets of GRASP meant to support the establishment of a security and transport substrate, described in the following section.

2.5.2. Discovery Unsolicited Link-Local (DULL) GRASP

Some services may need to use insecure GRASP discovery, response and flood messages without being able to use pre-existing security associations, for example as part of discovery for establishing security associations such as a security substrate for GRASP.

Such operations being intrinsically insecure, they need to be confined to link-local use to minimize the risk of malicious actions. Possible examples include discovery of candidate ACP neighbors [I-D.ietf-anima-autonomic-control-plane], discovery of bootstrap proxies [I-D.ietf-anima-bootstrapping-keyinfra] or perhaps initialization services in networks using GRASP without being fully autonomic (e.g., no ACP). Such usage MUST be limited to link-local operations on a single interface and MUST be confined to a separate insecure instance of GRASP with its own copy of all GRASP data structures. This instance is nicknamed DULL - Discovery Unsolicited Link-Local.

The detailed rules for the DULL instance of GRASP are as follows:

- An initiator MAY send Discovery or Flood Synchronization link-local multicast messages which MUST have a loop count of 1, to prevent off-link operations. Other unsolicited GRASP message types MUST NOT be sent.

- A responder MUST silently discard any message whose loop count is not 1.

- A responder MUST silently discard any message referring to a GRASP Objective that is not directly part of a service that requires this insecure mode.

- A responder MUST NOT relay any multicast messages.

- A Discovery Response MUST indicate a link-local address.

- A Discovery Response MUST NOT include a Divert option.

- o A node MUST silently discard any message whose source address is not link-local.

To minimize traffic possibly observed by third parties, GRASP traffic SHOULD be minimized by using only Flood Synchronization to announce objectives and their associated locators, rather than by using Discovery and Response. Further details are out of scope for this document

2.5.3. Transport Layer Usage

All GRASP messages, after they are serialized as a CBOR byte string, are transmitted as such directly over the transport protocol in use. The transport protocol(s) for a GRASP domain are specified by the security and transport substrate as introduced in Section 2.5.1.

GRASP discovery and flooding messages are designed for GRASP domain wide flooding through hop-by-hop link-local multicast forwarding between adjacent GRASP nodes. The GRASP security and transport substrate needs to specify how these link local multicasts are transported. This can be unreliable transport (UDP) but it SHOULD be reliable transport (e.g., TCP).

If the substrate specifies an unreliable transport such as UDP for discovery and flooding messages, then it MUST NOT use IP fragmentation because of its loss characteristic, especially in multi-hop flooding. GRASP MUST then enforce at the user API level a limit to the size of discovery and flooding messages, so that no fragmentation can occur. For IPv6 transport this means that those messages must be at most 1280 bytes sized IPv6 packets (unless there is a known larger minimum link MTU across the whole GRASP domain).

All other GRASP messages are unicast beteween group members of the GRASP domain. These MUST use a reliable transport protocol because GRASP itself does not provide for error detection, retransmission or flow control. Unless otherwise specified by the security and transport substrate, TCP MUST be used.

The security and transport substrate for GRASP in the ANI is the ACP. Unless otherwise noted, we assume this security and transport substrate in the remainder of this document when describing GRASPs message transport. In the ACP, TCP is used for GRASP unicast messages. GRASP discovery and flooding messages also use TCP: These link-local messages are forwarded by replicating them to all adjacent GRASP nodes on the link via TCP connections to those adjacent GRASP nodes. Because of this, GRASP in the ANI has no limitations on the size of discovery and flooding messages with respect to fragmentation

issues. UDP is used in the ANI with GRASP only with DULL when the ACP is built to discover ACP/GRASP neighbors on links.

For link-local UDP multicast, the GRASP protocol listens to the well-known GRASP Listen Port (Section 2.6). Transport connections for Discovery and Flooding on relay nodes must terminate in GRASP instances (eg: GRASP ASAs) so that link-local multicast, hop-by-hop flooding of M_DISCOVERY and M_FLOOD and hop-by-hop forwarding of M_RESPONSE and caching of those responses along the path work correctly.

Unicast transport connections used for synchronization and negotiation can terminate directly in ASAs that implement objectives and therefore this traffic does not need to pass through GRASP instances. For this, the ASA listens on its own dynamically assigned ports, which are communicated to its peers during discovery. Alternatively, the GRASP instance can also terminate the unicast transport connections and pass the traffic from/to the ASA if that is preferrable in some implementation (eg: to better decouple ASAs from network connections).

2.5.4. Discovery Mechanism and Procedures

2.5.4.1. Separated discovery and negotiation mechanisms

Although discovery and negotiation or synchronization are defined together in GRASP, they are separate mechanisms. The discovery process could run independently from the negotiation or synchronization process. Upon receiving a Discovery (Section 2.8.4) message, the recipient node should return a response message in which it either indicates itself as a discovery responder or diverts the initiator towards another more suitable ASA. However, this response may be delayed if the recipient needs to relay the discovery onwards, as described below.

The discovery action (M_DISCOVERY) will normally be followed by a negotiation (M_REQ_NEG) or synchronization (M_REQ_SYN) action. The discovery results could be utilized by the negotiation protocol to decide which ASA the initiator will negotiate with.

The initiator of a discovery action for a given objective need not be capable of responding to that objective as a Negotiation Counterpart, as a Synchronization Responder or as source for flooding. For example, an ASA might perform discovery even if it only wishes to act a Synchronization Initiator or Negotiation Initiator. Such an ASA does not itself need to respond to discovery messages.

It is also entirely possible to use GRASP discovery without any subsequent negotiation or synchronization action. In this case, the discovered objective is simply used as a name during the discovery process and any subsequent operations between the peers are outside the scope of GRASP.

2.5.4.2. Discovery Overview

A complete discovery process will start with a multicast (of M_DISCOVERY) on the local link. On-link neighbors supporting the discovery objective will respond directly (with M_RESPONSE). A neighbor with multiple interfaces may respond with a cached discovery response. If it has no cached response, it will relay the discovery on its other GRASP interfaces. If a node receiving the relayed discovery supports the discovery objective, it will respond to the relayed discovery. If it has a cached response, it will respond with that. If not, it will repeat the discovery process, which thereby becomes iterative. The loop count and timeout will ensure that the process ends. Further details are given below.

A Discovery message MAY be sent unicast to a peer node, which SHOULD then proceed exactly as if the message had been multicast, except that when TCP is used, the response will be on the same socket as the query. However, this mode does not guarantee successful discovery in the general case.

2.5.4.3. Discovery Procedures

Discovery starts as an on-link operation. The Divert option can tell the discovery initiator to contact an off-link ASA for that discovery objective. If the security and transport substrate of the GRASP domain (see Section 2.5.3) uses UDP link-local multicast then the discovery initiator sends these to the ALL_GRASP_NEIGHBORS link-local multicast address (Section 2.6) and and all GRASP nodes need to listen to this address to act as discovery responder. Because this port is unique in a device, this is a function of the GRASP instance and not of an individual ASA. As a result, each ASA will need to register the objectives that it supports with the local GRASP instance.

If an ASA in a neighbor device supports the requested discovery objective, the device SHOULD respond to the link-local multicast with a unicast Discovery Response message (Section 2.8.5) with locator option(s), unless it is temporarily unavailable. Otherwise, if the neighbor has cached information about an ASA that supports the requested discovery objective (usually because it discovered the same objective before), it SHOULD respond with a Discovery Response message with a Divert option pointing to the appropriate Discovery

Responder. However, it SHOULD NOT respond with a cached response on an interface if it learnt that information from the same interface, because the peer in question will answer directly if still operational.

If a device has no information about the requested discovery objective, and is not acting as a discovery relay (see below) it MUST silently discard the Discovery message.

The discovery initiator MUST set a reasonable timeout on the discovery process. A suggested value is 100 milliseconds multiplied by the loop count embedded in the objective.

If no discovery response is received within the timeout, the Discovery message MAY be repeated, with a newly generated Session ID (Section 2.7). An exponential backoff SHOULD be used for subsequent repetitions, to limit the load during busy periods. The details of the backoff algorithm will depend on the use case for the objective concerned but MUST be consistent with the recommendations in [RFC8085] for low data-volume multicast. Frequent repetition might be symptomatic of a denial of service attack.

After a GRASP device successfully discovers a locator for a Discovery Responder supporting a specific objective, it SHOULD cache this information, including the interface index [RFC3493] via which it was discovered. This cache record MAY be used for future negotiation or synchronization, and the locator SHOULD be passed on when appropriate as a Divert option to another Discovery Initiator.

The cache mechanism MUST include a lifetime for each entry. The lifetime is derived from a time-to-live (ttl) parameter in each Discovery Response message. Cached entries MUST be ignored or deleted after their lifetime expires. In some environments, unplanned address renumbering might occur. In such cases, the lifetime SHOULD be short compared to the typical address lifetime. The discovery mechanism needs to track the node's current address to ensure that Discovery Responses always indicate the correct address.

If multiple Discovery Responders are found for the same objective, they SHOULD all be cached, unless this creates a resource shortage. The method of choosing between multiple responders is an implementation choice. This choice MUST be available to each ASA but the GRASP implementation SHOULD provide a default choice.

Because Discovery Responders will be cached in a finite cache, they might be deleted at any time. In this case, discovery will need to be repeated. If an ASA exits for any reason, its locator might still

be cached for some time, and attempts to connect to it will fail. ASAs need to be robust in these circumstances.

2.5.4.4. Discovery Relaying

A GRASP instance with multiple link-layer interfaces (typically running in a router) MUST support discovery on all GRASP interfaces. We refer to this as a 'relaying instance'.

DULL Instances (Section 2.5.2) are always single-interface instances and therefore MUST NOT perform discovery relaying.

If a relaying instance receives a Discovery message on a given interface for a specific objective that it does not support and for which it has not previously cached a Discovery Responder, it MUST relay the query by re-issuing a new Discovery message as a link-local multicast on its other GRASP interfaces.

The relayed discovery message MUST have the same Session ID and Initiator field as the incoming (see Section 2.8.4). The Initiator IP address field is only used to allow for disambiguation of the Session ID and is never used to address Response packets. Response packets are sent back to the relaying instance, not the original initiator.

The M_DISCOVERY message does not encode the transport address of the originator or relay. Response packets must therefore be sent to the transport layer address of the connection on which the M_DISCOVERY message was received. If the M_DISCOVERY was relayed via a reliable hop-by-hop transport connection, the response is simply sent back via the same connection.

If the M_DISCOVERY was relayed via link-local (eg: UDP) multicast, the response is sent back via a reliable hop-by-hop transport connection with the same port number as the source port of the link-local multicast. Therefore, if link-local multicast is used and M_RESPONSE messages are required (which is the case in almost all GRASP instances except for the limited use of DULL instances in the ANI), GRASP needs to be able to bind to one port number on UDP from which to originate the link-local multicast M_DISCOVERY messages and the same port number on the reliable hop-by-hop transport (eg: TCP by default) to be able to respond to transport connections from responders that want to send M_RESPONSE messages back. Note that this port does not need to be the GRASP_LISTEN_PORT.

The relaying instance MUST decrement the loop count within the objective, and MUST NOT relay the Discovery message if the result is zero. Also, it MUST limit the total rate at which it relays

discovery messages to a reasonable value, in order to mitigate possible denial of service attacks. For example, the rate limit could be set to a small multiple of the observed rate of discovery messages during normal operation. The relaying instance MUST cache the Session ID value and initiator address of each relayed Discovery message until any Discovery Responses have arrived or the discovery process has timed out. To prevent loops, it MUST NOT relay a Discovery message which carries a given cached Session ID and initiator address more than once. These precautions avoid discovery loops and mitigate potential overload.

Since the relay device is unaware of the timeout set by the original initiator it SHOULD set a suitable timeout for the relayed discovery. A suggested value is 100 milliseconds multiplied by the remaining loop count.

The discovery results received by the relaying instance MUST in turn be sent as a Discovery Response message to the Discovery message that caused the relay action.

2.5.4.5. Rapid Mode (Discovery with Negotiation or Synchronization)

A Discovery message MAY include an Objective option. This allows a rapid mode of negotiation (Section 2.5.5.1) or synchronization (Section 2.5.6.3). Rapid mode is currently limited to a single objective for simplicity of design and implementation. A possible future extension is to allow multiple objectives in rapid mode for greater efficiency.

2.5.5. Negotiation Procedures

A negotiation initiator opens a transport connection to a counterpart ASA using the address, protocol and port obtained during discovery. It then sends a negotiation request (using M_REQ_NEG) to the counterpart, including a specific negotiation objective. It may request the negotiation counterpart to make a specific configuration. Alternatively, it may request a certain simulation or forecast result by sending a dry run configuration. The details, including the distinction between a dry run and a live configuration change, will be defined separately for each type of negotiation objective. Any state associated with a dry run operation, such as temporarily reserving a resource for subsequent use in a live run, is entirely a matter for the designer of the ASA concerned.

Each negotiation session as a whole is subject to a timeout (default GRASP_DEF_TIMEOUT milliseconds, Section 2.6), initialised when the request is sent (see Section 2.8.6). If no reply message of any kind is received within the timeout, the negotiation request MAY be

repeated, with a newly generated Session ID (Section 2.7). An exponential backoff SHOULD be used for subsequent repetitions. The details of the backoff algorithm will depend on the use case for the objective concerned.

If the counterpart can immediately apply the requested configuration, it will give an immediate positive (O_ACCEPT) answer (using M_END). This will end the negotiation phase immediately. Otherwise, it will negotiate (using M_NEGOTIATE). It will reply with a proposed alternative configuration that it can apply (typically, a configuration that uses fewer resources than requested by the negotiation initiator). This will start a bi-directional negotiation (using M_NEGOTIATE) to reach a compromise between the two ASAs.

The negotiation procedure is ended when one of the negotiation peers sends a Negotiation Ending (M_END) message, which contains an accept (O_ACCEPT) or decline (O_DECLINE) option and does not need a response from the negotiation peer. Negotiation may also end in failure (equivalent to a decline) if a timeout is exceeded or a loop count is exceeded. When the procedure ends for whatever reason, the transport connection SHOULD be closed. A transport session failure is treated as a negotiation failure.

A negotiation procedure concerns one objective and one counterpart. Both the initiator and the counterpart may take part in simultaneous negotiations with various other ASAs, or in simultaneous negotiations about different objectives. Thus, GRASP is expected to be used in a multi-threaded mode or its logical equivalent. Certain negotiation objectives may have restrictions on multi-threading, for example to avoid over-allocating resources.

Some configuration actions, for example wavelength switching in optical networks, might take considerable time to execute. The ASA concerned needs to allow for this by design, but GRASP does allow for a peer to insert latency in a negotiation process if necessary (Section 2.8.9, M_WAIT).

2.5.5.1. Rapid Mode (Discovery/Negotiation Linkage)

A Discovery message MAY include a Negotiation Objective option. In this case it is as if the initiator sent the sequence M_DISCOVERY, immediately followed by M_REQ_NEG. This has implications for the construction of the GRASP core, as it must carefully pass the contents of the Negotiation Objective option to the ASA so that it may evaluate the objective directly. When a Negotiation Objective option is present the ASA replies with an M_NEGOTIATE message (or M_END with O_ACCEPT if it is immediately satisfied with the

proposal), rather than with an M_RESPONSE. However, if the recipient node does not support rapid mode, discovery will continue normally.

It is possible that a Discovery Response will arrive from a responder that does not support rapid mode, before such a Negotiation message arrives. In this case, rapid mode will not occur.

This rapid mode could reduce the interactions between nodes so that a higher efficiency could be achieved. However, a network in which some nodes support rapid mode and others do not will have complex timing-dependent behaviors. Therefore, the rapid negotiation function SHOULD be disabled by default.

2.5.6. Synchronization and Flooding Procedures

2.5.6.1. Unicast Synchronization

A synchronization initiator opens a transport connection to a counterpart ASA using the address, protocol and port obtained during discovery. It then sends a synchronization request (using M_REQ_SYN) to the counterpart, including a specific synchronization objective. The counterpart responds with a Synchronization message (M_SYNCH, Section 2.8.10) containing the current value of the requested synchronization objective. No further messages are needed and the transport connection SHOULD be closed. A transport session failure is treated as a synchronization failure.

If no reply message of any kind is received within a given timeout (default GRASP_DEF_TIMEOUT milliseconds, Section 2.6), the synchronization request MAY be repeated, with a newly generated Session ID (Section 2.7). An exponential backoff SHOULD be used for subsequent repetitions. The details of the backoff algorithm will depend on the use case for the objective concerned.

2.5.6.2. Flooding

In the case just described, the message exchange is unicast and concerns only one synchronization objective. For large groups of nodes requiring the same data, synchronization flooding is available. For this, a flooding initiator MAY send an unsolicited Flood Synchronization message containing one or more Synchronization Objective option(s), if and only if the specification of those objectives permits it. This is sent as a multicast message to the ALL_GRASP_NEIGHBORS multicast address (Section 2.6).

Receiving flood multicasts is a function of the GRASP core, as in the case of discovery multicasts (Section 2.5.4.3).

To ensure that flooding does not result in a loop, the originator of the Flood Synchronization message MUST set the loop count in the objectives to a suitable value (the default is GRASP_DEF_LOOPCT). Also, a suitable mechanism is needed to avoid excessive multicast traffic. This mechanism MUST be defined as part of the specification of the synchronization objective(s) concerned. It might be a simple rate limit or a more complex mechanism such as the Trickle algorithm [RFC6206].

A GRASP device with multiple link-layer interfaces (typically a router) MUST support synchronization flooding on all GRASP interfaces. If it receives a multicast Flood Synchronization message on a given interface, it MUST relay it by re-issuing a Flood Synchronization message as a link-local multicast on its other GRASP interfaces. The relayed message MUST have the same Session ID as the incoming message and MUST be tagged with the IP address of its original initiator.

Link-layer Flooding is supported by GRASP by setting the loop count to 1, and sending with a link-local source address. Floods with link-local source addresses and a loop count other than 1 are invalid, and such messages MUST be discarded.

The relaying device MUST decrement the loop count within the first objective, and MUST NOT relay the Flood Synchronization message if the result is zero. Also, it MUST limit the total rate at which it relays Flood Synchronization messages to a reasonable value, in order to mitigate possible denial of service attacks. For example, the rate limit could be set to a small multiple of the observed rate of flood messages during normal operation. The relaying device MUST cache the Session ID value and initiator address of each relayed Flood Synchronization message for a time not less than twice GRASP_DEF_TIMEOUT milliseconds. To prevent loops, it MUST NOT relay a Flood Synchronization message which carries a given cached Session ID and initiator address more than once. These precautions avoid synchronization loops and mitigate potential overload.

Note that this mechanism is unreliable in the case of sleeping nodes, or new nodes that join the network, or nodes that rejoin the network after a fault. An ASA that initiates a flood SHOULD repeat the flood at a suitable frequency, which MUST be consistent with the recommendations in [RFC8085] for low data-volume multicast. The ASA SHOULD also act as a synchronization responder for the objective(s) concerned. Thus nodes that require an objective subject to flooding can either wait for the next flood or request unicast synchronization for that objective.

The multicast messages for synchronization flooding are subject to the security rules in Section 2.5.1. In practice this means that they MUST NOT be transmitted and MUST be ignored on receipt unless there is an operational ACP or equivalent strong security in place. However, because of the security weakness of link-local multicast (Section 4), synchronization objectives that are flooded SHOULD NOT contain unencrypted private information and SHOULD be validated by the recipient ASA.

2.5.6.3. Rapid Mode (Discovery/Synchronization Linkage)

A Discovery message MAY include a Synchronization Objective option. In this case the Discovery message also acts as a Request Synchronization message to indicate to the Discovery Responder that it could directly reply to the Discovery Initiator with a Synchronization message Section 2.8.10 with synchronization data for rapid processing, if the discovery target supports the corresponding synchronization objective. The design implications are similar to those discussed in Section 2.5.5.1.

It is possible that a Discovery Response will arrive from a responder that does not support rapid mode, before such a Synchronization message arrives. In this case, rapid mode will not occur.

This rapid mode could reduce the interactions between nodes so that a higher efficiency could be achieved. However, a network in which some nodes support rapid mode and others do not will have complex timing-dependent behaviors. Therefore, the rapid synchronization function SHOULD be configured off by default and MAY be configured on or off by Intent.

2.6. GRASP Constants

- ALL_GRASP_NEIGHBORS

  A link-local scope multicast address used by a GRASP-enabled device to discover GRASP-enabled neighbor (i.e., on-link) devices. All devices that support GRASP are members of this multicast group.

  * IPv6 multicast address: TBD1

  * IPv4 multicast address: TBD2

- GRASP_LISTEN_PORT (TBD3)

  A well-known UDP user port that every GRASP-enabled network device MUST listen to for link-local multicasts when UDP is used for

M_DISCOVERY or M_FLOOD messages in the GRASP instance This user port MAY also be used to listen for TCP or UDP unicast messages in a simple implementation of GRASP (Section 2.5.3).

- GRASP_DEF_TIMEOUT (60000 milliseconds)

  The default timeout used to determine that an operation has failed to complete.

- GRASP_DEF_LOOPCT (6)

  The default loop count used to determine that a negotiation has failed to complete, and to avoid looping messages.

- GRASP_DEF_MAX_SIZE (2048)

  The default maximum message size in bytes.

2.7. Session Identifier (Session ID)

This is an up to 32-bit opaque value used to distinguish multiple sessions between the same two devices. A new Session ID MUST be generated by the initiator for every new Discovery, Flood Synchronization or Request message. All responses and follow-up messages in the same discovery, synchronization or negotiation procedure MUST carry the same Session ID.

The Session ID SHOULD have a very low collision rate locally. It MUST be generated by a pseudo-random number generator (PRNG) using a locally generated seed which is unlikely to be used by any other device in the same network. The PRNG SHOULD be cryptographically strong [RFC4086]. When allocating a new Session ID, GRASP MUST check that the value is not already in use and SHOULD check that it has not been used recently, by consulting a cache of current and recent sessions. In the unlikely event of a clash, GRASP MUST generate a new value.

However, there is a finite probability that two nodes might generate the same Session ID value. For that reason, when a Session ID is communicated via GRASP, the receiving node MUST tag it with the initiator's IP address to allow disambiguation. In the highly unlikely event of two peers opening sessions with the same Session ID value, this tag will allow the two sessions to be distinguished. Multicast GRASP messages and their responses, which may be relayed between links, therefore include a field that carries the initiator's global IP address.

There is a highly unlikely race condition in which two peers start simultaneous negotiation sessions with each other using the same Session ID value. Depending on various implementation choices, this might lead to the two sessions being confused. See Section 2.8.6 for details of how to avoid this.

2.8. GRASP Messages

2.8.1. Message Overview

This section defines the GRASP message format and message types. Message types not listed here are reserved for future use.

The messages currently defined are:

Discovery and Discovery Response (M_DISCOVERY, M_RESPONSE).

Request Negotiation, Negotiation, Confirm Waiting and Negotiation End (M_REQ_NEG, M_NEGOTIATE, M_WAIT, M_END).

Request Synchronization, Synchronization, and Flood Synchronization (M_REQ_SYN, M_SYNCH, M_FLOOD.

No Operation and Invalid (M_NOOP, M_INVALID).

2.8.2. GRASP Message Format

GRASP messages share an identical header format and a variable format area for options. GRASP message headers and options are transmitted in Concise Binary Object Representation (CBOR) [RFC7049]. In this specification, they are described using CBOR data definition language (CDDL) [I-D.greevenbosch-appsawg-cbor-cddl]. Fragmentary CDDL is used to describe each item in this section. A complete and normative CDDL specification of GRASP is given in Section 5, including constants such as message types.

Every GRASP message, except the No Operation message, carries a Session ID (Section 2.7). Options are then presented serially in the options field.

In fragmentary CDDL, every GRASP message follows the pattern:

```
grasp-message = (message .within message-structure) / noop-message

message-structure = [MESSAGE_TYPE, session-id, ?initiator,
                     *grasp-option]

MESSAGE_TYPE = 1..255
session-id = 0..4294967295 ;up to 32 bits
grasp-option = any
```

The MESSAGE_TYPE indicates the type of the message and thus defines the expected options. Any options received that are not consistent with the MESSAGE_TYPE SHOULD be silently discarded.

The No Operation (noop) message is described in Section 2.8.13.

The various MESSAGE_TYPE values are defined in Section 5.

All other message elements are described below and formally defined in Section 5.

If an unrecognized MESSAGE_TYPE is received in a unicast message, an Invalid message (Section 2.8.12) MAY be returned. Otherwise the message MAY be logged and MUST be discarded. If an unrecognized MESSAGE_TYPE is received in a multicast message, it MAY be logged and MUST be silently discarded.

2.8.3. Message Size

GRASP nodes MUST be able to receive unicast messages of at least GRASP_DEF_MAX_SIZE bytes. GRASP nodes MUST NOT send unicast messages longer than GRASP_DEF_MAX_SIZE bytes unless a longer size is explicitly allowed for the objective concerned. For example, GRASP negotiation itself could be used to agree on a longer message size.

The message parser used by GRASP should be configured to know about the GRASP_DEF_MAX_SIZE, or any larger negotiated message size, so that it may defend against overly long messages.

The maximum size of multicast messages (M_DISCOVERY and M_FLOOD) depends on the link layer technology or link adaptation layer in use.

2.8.4. Discovery Message

In fragmentary CDDL, a Discovery message follows the pattern:

```
discovery-message = [M_DISCOVERY, session-id, initiator, objective]
```

A discovery initiator sends a Discovery message to initiate a discovery process for a particular objective option.

The discovery initiator sends all Discovery messages via UDP to port GRASP_LISTEN_PORT at the link-local ALL_GRASP_NEIGHBORS multicast address on each link-layer interface in use by GRASP. It then listens for unicast TCP responses on a given port, and stores the discovery results (including responding discovery objectives and corresponding unicast locators).

The listening port used for TCP MUST be the same port as used for sending the Discovery UDP multicast, on a given interface. In an implementation with a single GRASP instance in a node this MAY be GRASP_LISTEN_PORT. To support multiple instances in the same node, the GRASP discovery mechanism in each instance needs to find, for each interface, a dynamic port that it can bind to for both sending UDP link-local multicast and listening for TCP, before initiating any discovery.

The 'initiator' field in the message is a globally unique IP address of the initiator, for the sole purpose of disambiguating the Session ID in other nodes. If for some reason the initiator does not have a globally unique IP address, it MUST use a link-local address for this purpose that is highly likely to be unique, for example using [RFC7217]. Determination of a node's globally unique IP address is implementation-dependent.

A Discovery message MUST include exactly one of the following:

- a discovery objective option (Section 2.10.1). Its loop count MUST be set to a suitable value to prevent discovery loops (default value is GRASP_DEF_LOOPCT). If the discovery initiator requires only on-link responses, the loop count MUST be set to 1.

- a negotiation objective option (Section 2.10.1). This is used both for the purpose of discovery and to indicate to the discovery target that it MAY directly reply to the discovery initiatior with a Negotiation message for rapid processing, if it could act as the corresponding negotiation counterpart. The sender of such a Discovery message MUST initialize a negotiation timer and loop count in the same way as a Request Negotiation message (Section 2.8.6).

- a synchronization objective option (Section 2.10.1). This is used both for the purpose of discovery and to indicate to the discovery target that it MAY directly reply to the discovery initiator with a Synchronization message for rapid processing, if it could act as the corresponding synchronization counterpart. Its loop count

MUST be set to a suitable value to prevent discovery loops (default value is GRASP_DEF_LOOPCT).

As mentioned in Section 2.5.4.2, a Discovery message MAY be sent unicast to a peer node, which SHOULD then proceed exactly as if the message had been multicast.

2.8.5. Discovery Response Message

In fragmentary CDDL, a Discovery Response message follows the pattern:

```
response-message = [M_RESPONSE, session-id, initiator, ttl,
                    (+locator-option // divert-option), ?objective)]

ttl = 0..4294967295 ; in milliseconds
```

A node which receives a Discovery message SHOULD send a Discovery Response message if and only if it can respond to the discovery.

It MUST contain the same Session ID and initiator as the Discovery message.

It MUST contain a time-to-live (ttl) for the validity of the response, given as a positive integer value in milliseconds. Zero implies a value significantly greater than GRASP_DEF_TIMEOUT milliseconds (Section 2.6). A suggested value is ten times that amount.

It MAY include a copy of the discovery objective from the Discovery message.

It is sent to the sender of the Discovery message via TCP at the port used to send the Discovery message (as explained in Section 2.8.4). In the case of a relayed Discovery message, the Discovery Response is thus sent to the relay, not the original initiator.

In all cases, the transport session SHOULD be closed after sending the Discovery Response. A transport session failure is treated as no response.

If the responding node supports the discovery objective of the discovery, it MUST include at least one kind of locator option (Section 2.9.5) to indicate its own location. A sequence of multiple kinds of locator options (e.g. IP address option and FQDN option) is also valid.

If the responding node itself does not support the discovery objective, but it knows the locator of the discovery objective, then it SHOULD respond to the discovery message with a divert option (Section 2.9.2) embedding a locator option or a combination of multiple kinds of locator options which indicate the locator(s) of the discovery objective.

More details on the processing of Discovery Responses are given in Section 2.5.4.

2.8.6. Request Messages

In fragmentary CDDL, Request Negotiation and Request Synchronization messages follow the patterns:

```
request-negotiation-message = [M_REQ_NEG, session-id, objective]

request-synchronization-message = [M_REQ_SYN, session-id, objective]
```

A negotiation or synchronization requesting node sends the appropriate Request message to the unicast address of the negotiation or synchronization counterpart, using the appropriate protocol and port numbers (selected from the discovery result). If the discovery result is an FQDN, it will be resolved first.

A Request message MUST include the relevant objective option. In the case of Request Negotiation, the objective option MUST include the requested value.

When an initiator sends a Request Negotiation message, it MUST initialize a negotiation timer for the new negotiation thread. The default is GRASP_DEF_TIMEOUT milliseconds. Unless this timeout is modified by a Confirm Waiting message (Section 2.8.9), the initiator will consider that the negotiation has failed when the timer expires.

Similarly, when an initiator sends a Request Synchronization, it SHOULD initialize a synchronization timer. The default is GRASP_DEF_TIMEOUT milliseconds. The initiator will consider that synchronization has failed if there is no response before the timer expires.

When an initiator sends a Request message, it MUST initialize the loop count of the objective option with a value defined in the specification of the option or, if no such value is specified, with GRASP_DEF_LOOPCT.

If a node receives a Request message for an objective for which no ASA is currently listening, it MUST immediately close the relevant socket to indicate this to the initiator. This is to avoid unnecessary timeouts if, for example, an ASA exits prematurely but the GRASP core is listening on its behalf.

To avoid the highly unlikely race condition in which two nodes simultaneously request sessions with each other using the same Session ID (Section 2.7), when a node receives a Request message, it MUST verify that the received Session ID is not already locally active. In case of a clash, it MUST discard the Request message, in which case the initiator will detect a timeout.

2.8.7. Negotiation Message

In fragmentary CDDL, a Negotiation message follows the pattern:

```
negotiate-message = [M_NEGOTIATE, session-id, objective]
```

A negotiation counterpart sends a Negotiation message in response to a Request Negotiation message, a Negotiation message, or a Discovery message in Rapid Mode. A negotiation process MAY include multiple steps.

The Negotiation message MUST include the relevant Negotiation Objective option, with its value updated according to progress in the negotiation. The sender MUST decrement the loop count by 1. If the loop count becomes zero the message MUST NOT be sent. In this case the negotiation session has failed and will time out.

2.8.8. Negotiation End Message

In fragmentary CDDL, a Negotiation End message follows the pattern:

```
end-message = [M_END, session-id, accept-option / decline-option]
```

A negotiation counterpart sends an Negotiation End message to close the negotiation. It MUST contain either an accept or a decline option, defined in Section 2.9.3 and Section 2.9.4. It could be sent either by the requesting node or the responding node.

2.8.9. Confirm Waiting Message

In fragmentary CDDL, a Confirm Waiting message follows the pattern:

```
wait-message = [M_WAIT, session-id, waiting-time]
waiting-time = 0..4294967295 ; in milliseconds
```

A responding node sends a Confirm Waiting message to ask the requesting node to wait for a further negotiation response. It might be that the local process needs more time or that the negotiation depends on another triggered negotiation. This message MUST NOT include any other options. When received, the waiting time value overwrites and restarts the current negotiation timer (Section 2.8.6).

The responding node SHOULD send a Negotiation, Negotiation End or another Confirm Waiting message before the negotiation timer expires. If not, when the initiator's timer expires, the initiator MUST treat the negotiation procedure as failed.

2.8.10. Synchronization Message

In fragmentary CDDL, a Synchronization message follows the pattern:

```
synch-message = [M_SYNCH, session-id, objective]
```

A node which receives a Request Synchronization, or a Discovery message in Rapid Mode, sends back a unicast Synchronization message with the synchronization data, in the form of a GRASP Option for the specific synchronization objective present in the Request Synchronization.

2.8.11. Flood Synchronization Message

In fragmentary CDDL, a Flood Synchronization message follows the pattern:

```
flood-message = [M_FLOOD, session-id, initiator, ttl,
                +[objective, (locator-option / [])]]

ttl = 0..4294967295 ; in milliseconds
```

A node MAY initiate flooding by sending an unsolicited Flood Synchronization Message with synchronization data. This MAY be sent to port GRASP_LISTEN_PORT at the link-local ALL_GRASP_NEIGHBORS multicast address, in accordance with the rules in Section 2.5.6.

The initiator address is provided, as described for Discovery messages (Section 2.8.4), only to disambiguate the Session ID.

The message MUST contain a time-to-live (ttl) for the validity of the contents, given as a positive integer value in milliseconds. There is no default; zero indicates an indefinite lifetime.

The synchronization data are in the form of GRASP Option(s) for specific synchronization objective(s). The loop count(s) MUST be set to a suitable value to prevent flood loops (default value is GRASP_DEF_LOOPCT).

Each objective option MAY be followed by a locator option associated with the flooded objective. In its absence, an empty option MUST be included to indicate a null locator.

A node that receives a Flood Synchronization message MUST cache the received objectives for use by local ASAs. Each cached objective MUST be tagged with the locator option sent with it, or with a null tag if an empty locator option was sent. If a subsequent Flood Synchronization message carrying an objective with same name and the same tag, the corresponding cached copy of the objective MUST be overwritten. If a subsequent Flood Synchronization message carrying an objective with same name arrives with a different tag, a new cached entry MUST be created.

Note: the purpose of this mechanism is to allow the recipient of flooded values to distinguish between different senders of the same objective, and if necessary communicate with them using the locator, protocol and port included in the locator option. Many objectives will not need this mechanism, so they will be flooded with a null locator.

Cached entries MUST be ignored or deleted after their lifetime expires.

2.8.12. Invalid Message

In fragmentary CDDL, an Invalid message follows the pattern:

```
invalid-message = [M_INVALID, session-id, ?any]
```

This message MAY be sent by an implementation in response to an incoming unicast message that it considers invalid. The session-id MUST be copied from the incoming message. The content SHOULD be diagnostic information such as a partial copy of the invalid message up to the maximum message size. An M_INVALID message MAY be silently ignored by a recipient. However, it could be used in support of extensibility, since it indicates that the remote node does not support a new or obsolete message or option.

An M_INVALID message MUST NOT be sent in response to an M_INVALID message.

2.8.13. No Operation Message

In fragmentary CDDL, a No Operation message follows the pattern:

```
noop-message = [M_NOOP]
```

This message MAY be sent by an implementation that for practical reasons needs to initialize a socket. It MUST be silently ignored by a recipient.

2.9. GRASP Options

This section defines the GRASP options for the negotiation and synchronization protocol signaling. Additional options may be defined in the future.

2.9.1. Format of GRASP Options

GRASP options are CBOR objects that MUST start with an unsigned integer identifying the specific option type carried in this option. These option types are formally defined in Section 5. Apart from that the only format requirement is that each option MUST be a well-formed CBOR object. In general a CBOR array format is RECOMMENDED to limit overhead.

GRASP options may be defined to include encapsulated GRASP options.

2.9.2. Divert Option

The Divert option is used to redirect a GRASP request to another node, which may be more appropriate for the intended negotiation or synchronization. It may redirect to an entity that is known as a specific negotiation or synchronization counterpart (on-link or off-link) or a default gateway. The divert option MUST only be encapsulated in Discovery Response messages. If found elsewhere, it SHOULD be silently ignored.

A discovery initiator MAY ignore a Divert option if it only requires direct discovery responses.

In fragmentary CDDL, the Divert option follows the pattern:

```
divert-option = [O_DIVERT, +locator-option]
```

The embedded Locator Option(s) (Section 2.9.5) point to diverted destination target(s) in response to a Discovery message.

2.9.3. Accept Option

The accept option is used to indicate to the negotiation counterpart that the proposed negotiation content is accepted.

The accept option MUST only be encapsulated in Negotiation End messages. If found elsewhere, it SHOULD be silently ignored.

In fragmentary CDDL, the Accept option follows the pattern:

```
accept-option = [O_ACCEPT]
```

2.9.4. Decline Option

The decline option is used to indicate to the negotiation counterpart the proposed negotiation content is declined and end the negotiation process.

The decline option MUST only be encapsulated in Negotiation End messages. If found elsewhere, it SHOULD be silently ignored.

In fragmentary CDDL, the Decline option follows the pattern:

```
decline-option = [O_DECLINE, ?reason]
reason = text  ;optional UTF-8 error message
```

Note: there might be scenarios where an ASA wants to decline the proposed value and restart the negotiation process. In this case it is an implementation choice whether to send a Decline option or to continue with a Negotiate message, with an objective option that contains a null value, or one that contains a new value that might achieve convergence.

2.9.5. Locator Options

These locator options are used to present reachability information for an ASA, a device or an interface. They are Locator IPv6 Address Option, Locator IPv4 Address Option, Locator FQDN (Fully Qualified Domain Name) Option and URI (Uniform Resource Identifier) Option.

Since ASAs will normally run as independent user programs, locator options need to indicate the network layer locator plus the transport protocol and port number for reaching the target. For this reason, the Locator Options for IP addresses and FQDNs include this information explicitly. In the case of the URI Option, this information can be encoded in the URI itself.

Note: It is assumed that all locators used in locator options are in scope throughout the GRASP domain. As stated in Section 2.2, GRASP is not intended to work across disjoint addressing or naming realms.

2.9.5.1. Locator IPv6 address option

In fragmentary CDDL, the IPv6 address option follows the pattern:

```
ipv6-locator-option = [O_IPv6_LOCATOR, ipv6-address,
                       transport-proto, port-number]
ipv6-address = bytes .size 16

transport-proto = IPPROTO_TCP / IPPROTO_UDP
IPPROTO_TCP = 6
IPPROTO_UDP = 17
port-number = 0..65535
```

The content of this option is a binary IPv6 address followed by the protocol number and port number to be used.

Note 1: The IPv6 address MUST normally have global scope. However, during initialization, a link-local address MAY be used for specific objectives only (Section 2.5.2). In this case the corresponding Discovery Response message MUST be sent via the interface to which the link-local address applies.

Note 2: A link-local IPv6 address MUST NOT be used when this option is included in a Divert option.

Note 3: The IPPROTO values are taken from the existing IANA Protocol Numbers registry in order to specify TCP or UDP. If GRASP requires future values that are not in that registry, a new registry for values outside the range 0..255 will be needed.

2.9.5.2. Locator IPv4 address option

In fragmentary CDDL, the IPv4 address option follows the pattern:

```
ipv4-locator-option = [O_IPv4_LOCATOR, ipv4-address,
                       transport-proto, port-number]
ipv4-address = bytes .size 4
```

The content of this option is a binary IPv4 address followed by the protocol number and port number to be used.

Note: If an operator has internal network address translation for IPv4, this option MUST NOT be used within the Divert option.

2.9.5.3. Locator FQDN option

In fragmentary CDDL, the FQDN option follows the pattern:

```
fqdn-locator-option = [O_FQDN_LOCATOR, text,
                       transport-proto, port-number]
```

The content of this option is the Fully Qualified Domain Name of the target followed by the protocol number and port number to be used.

Note 1: Any FQDN which might not be valid throughout the network in question, such as a Multicast DNS name [RFC6762], MUST NOT be used when this option is used within the Divert option.

Note 2: Normal GRASP operations are not expected to use this option. It is intended for special purposes such as discovering external services.

2.9.5.4. Locator URI option

In fragmentary CDDL, the URI option follows the pattern:

```
uri-locator = [O_URI_LOCATOR, text,
               transport-proto / null, port-number / null]
```

The content of this option is the Uniform Resource Identifier of the target followed by the protocol number and port number to be used (or by null values if not required) [RFC3986].

Note 1: Any URI which might not be valid throughout the network in question, such as one based on a Multicast DNS name [RFC6762], MUST NOT be used when this option is used within the Divert option.

Note 2: Normal GRASP operations are not expected to use this option. It is intended for special purposes such as discovering external services. Therefore its use is not further described in this specification.

2.10. Objective Options

2.10.1. Format of Objective Options

An objective option is used to identify objectives for the purposes of discovery, negotiation or synchronization. All objectives MUST be in the following format, described in fragmentary CDDL:

```
objective = [objective-name, objective-flags, loop-count, ?objective-value]

objective-name = text
objective-value = any
loop-count = 0..255
```

All objectives are identified by a unique name which is a UTF-8 string [RFC3629], to be compared byte by byte.

The names of generic objectives MUST NOT include a colon (":") and MUST be registered with IANA (Section 6).

The names of privately defined objectives MUST include at least one colon (":"). The string preceding the last colon in the name MUST be globally unique and in some way identify the entity or person defining the objective. The following three methods MAY be used to create such a globally unique string:

1. The unique string is a decimal number representing a registered 32 bit Private Enterprise Number (PEN) [RFC5612] that uniquely identifies the enterprise defining the objective.

2. The unique string is a fully qualified domain name that uniquely identifies the entity or person defining the objective.

3. The unique string is an email address that uniquely identifies the entity or person defining the objective.

The GRASP protocol treats the objective name as an opaque string. For example, "EX1", "32473:EX1", "example.com:EX1", "example.org:EX1 and "user@example.org:EX1" would be five different objectives.

The 'objective-flags' field is described below.

The 'loop-count' field is used for terminating negotiation as described in Section 2.8.7. It is also used for terminating discovery as described in Section 2.5.4, and for terminating flooding as described in Section 2.5.6.2. It is placed in the objective rather than in the GRASP message format because, as far as the ASA is concerned, it is a property of the objective itself.

The 'objective-value' field is to express the actual value of a negotiation or synchronization objective. Its format is defined in the specification of the objective and may be a simple value or a data structure of any kind, as long as it can be represented in CBOR. It is optional because it is optional in a Discovery or Discovery Response message.

2.10.2. Objective flags

An objective may be relevant for discovery only, for discovery and negotiation, or for discovery and synchronization. This is expressed in the objective by logical flag bits:

```
objective-flags = uint .bits objective-flag
objective-flag = &(
F_DISC: 0    ; valid for discovery
F_NEG: 1     ; valid for negotiation
F_SYNCH: 2   ; valid for synchronization
F_NEG_DRY: 3 ; negotiation is dry-run
)
```

These bits are independent and may be combined appropriately, e.g. (F_DISC and F_SYNCH) or (F_DISC and F_NEG) or (F_DISC and F_NEG and F_NEG_DRY).

Note that for a given negotiation session, an objective must be either used for negotiation, or for dry-run negotiation. Mixing the two modes in a single negotiation is not possible.

2.10.3. General Considerations for Objective Options

As mentioned above, Objective Options MUST be assigned a unique name. As long as privately defined Objective Options obey the rules above, this document does not restrict their choice of name, but the entity or person concerned SHOULD publish the names in use.

Names are expressed as UTF-8 strings for convenience in designing Objective Options for localized use. For generic usage, names expressed in the ASCII subset of UTF-8 are RECOMMENDED. Designers planning to use non-ASCII names are strongly advised to consult [RFC7564] or its successor to understand the complexities involved. Since the GRASP protocol compares names byte by byte, all issues of Unicode profiling and canonicalization MUST be specified in the design of the Objective Option.

All Objective Options MUST respect the CBOR patterns defined above as "objective" and MUST replace the "any" field with a valid CBOR data definition for the relevant use case and application.

An Objective Option that contains no additional fields beyond its "loop-count" can only be a discovery objective and MUST only be used in Discovery and Discovery Response messages.

The Negotiation Objective Options contain negotiation objectives, which vary according to different functions/services. They MUST be

carried by Discovery, Request Negotiation or Negotiation messages only. The negotiation initiator MUST set the initial "loop-count" to a value specified in the specification of the objective or, if no such value is specified, to GRASP_DEF_LOOPCT.

For most scenarios, there should be initial values in the negotiation requests. Consequently, the Negotiation Objective options MUST always be completely presented in a Request Negotiation message, or in a Discovery message in rapid mode. If there is no initial value, the value field SHOULD be set to the 'null' value defined by CBOR.

Synchronization Objective Options are similar, but MUST be carried by Discovery, Discovery Response, Request Synchronization, or Flood Synchronization messages only. They include value fields only in Synchronization or Flood Synchronization messages.

The design of an objective interacts in various ways with the design of the ASAs that will use it. ASA design considerations are discussed in [I-D.carpenter-anima-asa-guidelines].

2.10.4. Organizing of Objective Options

Generic objective options MUST be specified in documents available to the public and SHOULD be designed to use either the negotiation or the synchronization mechanism described above.

As noted earlier, one negotiation objective is handled by each GRASP negotiation thread. Therefore, a negotiation objective, which is based on a specific function or action, SHOULD be organized as a single GRASP option. It is NOT RECOMMENDED to organize multiple negotiation objectives into a single option, nor to split a single function or action into multiple negotiation objectives.

It is important to understand that GRASP negotiation does not support transactional integrity. If transactional integrity is needed for a specific objective, this must be ensured by the ASA. For example, an ASA might need to ensure that it only participates in one negotiation thread at the same time. Such an ASA would need to stop listening for incoming negotiation requests before generating an outgoing negotiation request.

A synchronization objective SHOULD be organized as a single GRASP option.

Some objectives will support more than one operational mode. An example is a negotiation objective with both a "dry run" mode (where the negotiation is to find out whether the other end can in fact make the requested change without problems) and a "live" mode, as

explained in Section 2.5.5. The semantics of such modes will be defined in the specification of the objectives. These objectives SHOULD include flags indicating the applicable mode(s).

An issue requiring particular attention is that GRASP itself is not a transactionally safe protocol. Any state associated with a dry run operation, such as temporarily reserving a resource for subsequent use in a live run, is entirely a matter for the designer of the ASA concerned.

As indicated in Section 2.1, an objective's value may include multiple parameters. Parameters might be categorized into two classes: the obligatory ones presented as fixed fields; and the optional ones presented in some other form of data structure embedded in CBOR. The format might be inherited from an existing management or configuration protocol, with the objective option acting as a carrier for that format. The data structure might be defined in a formal language, but that is a matter for the specifications of individual objectives. There are many candidates, according to the context, such as ABNF, RBNF, XML Schema, YANG, etc. The GRASP protocol itself is agnostic on these questions. The only restriction is that the format can be mapped into CBOR.

It is NOT RECOMMENDED to mix parameters that have significantly different response time characteristics in a single objective. Separate objectives are more suitable for such a scenario.

All objectives MUST support GRASP discovery. However, as mentioned in Section 2.3, it is acceptable for an ASA to use an alternative method of discovery.

Normally, a GRASP objective will refer to specific technical parameters as explained in Section 2.1. However, it is acceptable to define an abstract objective for the purpose of managing or coordinating ASAs. It is also acceptable to define a special-purpose objective for purposes such as trust bootstrapping or formation of the ACP.

To guarantee convergence, a limited number of rounds or a timeout is needed for each negotiation objective. Therefore, the definition of each negotiation objective SHOULD clearly specify this, for example a default loop count and timeout, so that the negotiation can always be terminated properly. If not, the GRASP defaults will apply.

There must be a well-defined procedure for concluding that a negotiation cannot succeed, and if so deciding what happens next (e.g., deadlock resolution, tie-breaking, or revert to best-effort

service). This MUST be specified for individual negotiation objectives.

2.10.5. Experimental and Example Objective Options

The names "EX0" through "EX9" have been reserved for experimental options. Multiple names have been assigned because a single experiment may use multiple options simultaneously. These experimental options are highly likely to have different meanings when used for different experiments. Therefore, they SHOULD NOT be used without an explicit human decision and MUST NOT be used in unmanaged networks such as home networks.

These names are also RECOMMENDED for use in documentation examples.

3. Implementation Status [RFC Editor: please remove]

Two prototype implementations of GRASP have been made.

3.1. BUPT C++ Implementation

- Name: BaseNegotiator.cpp, msg.cpp, Client.cpp, Server.cpp
- Description: C++ implementation of GRASP core and API
- Maturity: Prototype code, interoperable between Ubuntu.
- Coverage: Corresponds to draft-carpenter-anima-gdn-protocol-03. Since it was implemented based on the old version draft, the most significant limitations comparing to current protocol design include:
  - Not support CBOR
  - Not support Flooding
  - Not support loop avoidance
  - only coded for IPv6, any IPv4 is accidental
- Licensing: Huawei License.
- Experience: https://github.com/liubingpang/IETF-Anima-Signaling-Protocol/blob/master/README.md
- Contact: https://github.com/liubingpang/IETF-Anima-Signaling-Protocol

3.2. Python Implementation

- Name: graspy
- Description: Python 3 implementation of GRASP core and API.
- Maturity: Prototype code, interoperable between Windows 7 and Linux.
- Coverage: Corresponds to draft-ietf-anima-grasp-13. Limitations include:
  - insecure: uses a dummy ACP module
  - only coded for IPv6, any IPv4 is accidental
  - FQDN and URI locators incompletely supported
  - no code for rapid mode
  - relay code is lazy (no rate control)
  - all unicast transactions use TCP (no unicast UDP). Experimental code for unicast UDP proved to be complex and brittle.
  - optional Objective option in Response messages not implemented
  - workarounds for defects in Python socket module and Windows socket peculiarities
- Licensing: Simplified BSD
- Experience: Tested on Windows, Linux and MacOS. https://www.cs.auckland.ac.nz/~brian/graspy/graspy.pdf
- Contact: https://www.cs.auckland.ac.nz/~brian/graspy/

4. Security Considerations

A successful attack on negotiation-enabled nodes would be extremely harmful, as such nodes might end up with a completely undesirable configuration that would also adversely affect their peers. GRASP nodes and messages therefore require full protection. As explained in Section 2.5.1, GRASP MUST run within a secure environment such as the Autonomic Control Plane [I-D.ietf-anima-autonomic-control-plane], except for the constrained instances described in Section 2.5.2.

- Authentication

  A cryptographically authenticated identity for each device is needed in an autonomic network. It is not safe to assume that a large network is physically secured against interference or that all personnel are trustworthy. Each autonomic node MUST be capable of proving its identity and authenticating its messages. GRASP relies on a separate external certificate-based security mechanism to support authentication, data integrity protection, and anti-replay protection.

  Since GRASP must be deployed in an existing secure environment, the protocol itself specifies nothing concerning the trust anchor and certification authority. For example, in the Autonomic Control Plane [I-D.ietf-anima-autonomic-control-plane], all nodes can trust each other and the ASAs installed in them.

  If GRASP is used temporarily without an external security mechanism, for example during system bootstrap (Section 2.5.1), the Session ID (Section 2.7) will act as a nonce to provide limited protection against third parties injecting responses. A full analysis of the secure bootstrap process is in [I-D.ietf-anima-bootstrapping-keyinfra].

- Authorization and Roles

  The GRASP protocol is agnostic about the roles and capabilities of individual ASAs and about which objectives a particular ASA is authorized to support. An implementation might support precautions such as allowing only one ASA in a given node to modify a given objective, but this may not be appropriate in all cases. For example, it might be operationally useful to allow an old and a new version of the same ASA to run simultaneously during an overlap period. These questions are out of scope for the present specification.

- Privacy and confidentiality

  GRASP is intended for network management purposes involving network elements, not end hosts. Therefore, no personal information is expected to be involved in the signaling protocol, so there should be no direct impact on personal privacy. Nevertheless, applications that do convey personal information cannot be excluded. Also, traffic flow paths, VPNs, etc. could be negotiated, which could be of interest for traffic analysis. Operators generally want to conceal details of their network topology and traffic density from outsiders. Therefore, since insider attacks cannot be excluded in a large network, the

security mechanism for the protocol MUST provide message confidentiality. This is why Section 2.5.1 requires either an ACP or an alternative security mechanism.

- Link-local multicast security

  GRASP has no reasonable alternative to using link-local multicast for Discovery or Flood Synchronization messages and these messages are sent in clear and with no authentication. They are only sent on interfaces within the autonomic network (see Section 2.1 and Section 2.5.1). They are however available to on-link eavesdroppers, and could be forged by on-link attackers. In the case of Discovery, the Discovery Responses are unicast and will therefore be protected (Section 2.5.1), and an untrusted forger will not be able to receive responses. In the case of Flood Synchronization, an on-link eavesdropper will be able to receive the flooded objectives but there is no response message to consider. Some precautions for Flood Synchronization messages are suggested in Section 2.5.6.2.

- DoS Attack Protection

  GRASP discovery partly relies on insecure link-local multicast. Since routers participating in GRASP sometimes relay discovery messages from one link to another, this could be a vector for denial of service attacks. Some mitigations are specified in Section 2.5.4. However, malicious code installed inside the Autonomic Control Plane could always launch DoS attacks consisting of spurious discovery messages, or of spurious discovery responses. It is important that firewalls prevent any GRASP messages from entering the domain from an unknown source.

- Security during bootstrap and discovery

  A node cannot trust GRASP traffic from other nodes until the security environment (such as the ACP) has identified the trust anchor and can authenticate traffic by validating certificates for other nodes. Also, until it has succesfully enrolled [I-D.ietf-anima-bootstrapping-keyinfra] a node cannot assume that other nodes are able to authenticate its own traffic. Therefore, GRASP discovery during the bootstrap phase for a new device will inevitably be insecure. Secure synchronization and negotiation will be impossible until enrollment is complete. Further details are given in Section 2.5.2.

- Security of discovered locators

When GRASP discovery returns an IP address, it MUST be that of a node within the secure environment (Section 2.5.1). If it returns an FQDN or a URI, the ASA that receives it MUST NOT assume that the target of the locator is within the secure environment.

5. CDDL Specification of GRASP

```
<CODE BEGINS>
grasp-message = (message .within message-structure) / noop-message

message-structure = [MESSAGE_TYPE, session-id, ?initiator,
                     *grasp-option]

MESSAGE_TYPE = 0..255
session-id = 0..4294967295 ;up to 32 bits
grasp-option = any

message /= discovery-message
discovery-message = [M_DISCOVERY, session-id, initiator, objective]

message /= response-message ;response to Discovery
response-message = [M_RESPONSE, session-id, initiator, ttl,
                    (+locator-option // divert-option), ?objective]

message /= synch-message ;response to Synchronization request
synch-message = [M_SYNCH, session-id, objective]

message /= flood-message
flood-message = [M_FLOOD, session-id, initiator, ttl,
                 +[objective, (locator-option / [])]]

message /= request-negotiation-message
request-negotiation-message = [M_REQ_NEG, session-id, objective]

message /= request-synchronization-message
request-synchronization-message = [M_REQ_SYN, session-id, objective]

message /= negotiation-message
negotiation-message = [M_NEGOTIATE, session-id, objective]

message /= end-message
end-message = [M_END, session-id, accept-option / decline-option ]

message /= wait-message
wait-message = [M_WAIT, session-id, waiting-time]

message /= invalid-message
invalid-message = [M_INVALID, session-id, ?any]
```

```
noop-message = [M_NOOP]

divert-option = [O_DIVERT, +locator-option]

accept-option = [O_ACCEPT]

decline-option = [O_DECLINE, ?reason]
reason = text  ;optional UTF-8 error message

waiting-time = 0..4294967295 ; in milliseconds
ttl = 0..4294967295 ; in milliseconds

locator-option /= [O_IPv4_LOCATOR, ipv4-address,
                   transport-proto, port-number]
ipv4-address = bytes .size 4

locator-option /= [O_IPv6_LOCATOR, ipv6-address,
                   transport-proto, port-number]
ipv6-address = bytes .size 16

locator-option /= [O_FQDN_LOCATOR, text, transport-proto, port-number]

locator-option /= [O_URI_LOCATOR, text,
                   transport-proto / null, port-number / null]

transport-proto = IPPROTO_TCP / IPPROTO_UDP
IPPROTO_TCP = 6
IPPROTO_UDP = 17
port-number = 0..65535

initiator = ipv4-address / ipv6-address

objective-flags = uint .bits objective-flag

objective-flag = &(
  F_DISC: 0    ; valid for discovery
  F_NEG: 1     ; valid for negotiation
  F_SYNCH: 2   ; valid for synchronization
  F_NEG_DRY: 3 ; negotiation is dry-run
)

objective = [objective-name, objective-flags, loop-count, ?objective-value]

objective-name = text ;see section "Format of Objective Options"

objective-value = any

loop-count = 0..255
```

```
; Constants for message types and option types

M_NOOP = 0
M_DISCOVERY = 1
M_RESPONSE = 2
M_REQ_NEG = 3
M_REQ_SYN = 4
M_NEGOTIATE = 5
M_END = 6
M_WAIT = 7
M_SYNCH = 8
M_FLOOD = 9
M_INVALID = 99

O_DIVERT = 100
O_ACCEPT = 101
O_DECLINE = 102
O_IPv6_LOCATOR = 103
O_IPv4_LOCATOR = 104
O_FQDN_LOCATOR = 105
O_URI_LOCATOR = 106
<CODE ENDS>
```

6. IANA Considerations

This document defines the GeneRic Autonomic Signaling Protocol (GRASP).

Section 2.6 explains the following link-local multicast addresses, which IANA is requested to assign for use by GRASP:

ALL_GRASP_NEIGHBORS multicast address (IPv6): (TBD1). Assigned in the IPv6 Link-Local Scope Multicast Addresses registry.

ALL_GRASP_NEIGHBORS multicast address (IPv4): (TBD2). Assigned in the IPv4 Multicast Local Network Control Block.

Section 2.6 explains the following User Port, which IANA is requested to assign for use by GRASP for both UDP and TCP:

GRASP_LISTEN_PORT: (TBD3)
Service Name: Generic Autonomic Signaling Protocol (GRASP)
Transport Protocols: UDP, TCP
Assignee: iesg@ietf.org
Contact: chair@ietf.org
Description: See Section 2.6
Reference: RFC XXXX (this document)

The IANA is requested to create a GRASP Parameter Registry including two registry tables. These are the GRASP Messages and Options Table and the GRASP Objective Names Table.

GRASP Messages and Options Table. The values in this table are names paired with decimal integers. Future values MUST be assigned using the Standards Action policy defined by [RFC8126]. The following initial values are assigned by this document:

```
M_NOOP = 0
M_DISCOVERY = 1
M_RESPONSE = 2
M_REQ_NEG = 3
M_REQ_SYN = 4
M_NEGOTIATE = 5
M_END = 6
M_WAIT = 7
M_SYNCH = 8
M_FLOOD = 9
M_INVALID = 99

O_DIVERT = 100
O_ACCEPT = 101
O_DECLINE = 102
O_IPv6_LOCATOR = 103
O_IPv4_LOCATOR = 104
O_FQDN_LOCATOR = 105
O_URI_LOCATOR = 106
```

GRASP Objective Names Table. The values in this table are UTF-8 strings which MUST NOT include a colon (":"), according to Section 2.10.1. Future values MUST be assigned using the Specification Required policy defined by [RFC8126].

To assist expert review of a new objective, the specification should include a precise description of the format of the new objective, with sufficient explanation of its semantics to allow independent implementations. See Section 2.10.3 for more details. If the new objective is similar in name or purpose to a previously registered objective, the specification should explain why a new objective is justified.

The following initial values are assigned by this document:

EX0
EX1
EX2
EX3
EX4
EX5
EX6
EX7
EX8
EX9

7. Acknowledgements

A major contribution to the original version of this document was made by Sheng Jiang and significant contributions were made by Toerless Eckert. Significant early review inputs were received from Joel Halpern, Barry Leiba, Charles E. Perkins, and Michael Richardson. William Atwood provided important assistance in debugging a prototype implementation.

Valuable comments were received from Michael Behringer, Jeferson Campos Nobre, Laurent Ciavaglia, Zongpeng Du, Yu Fu, Joel Jaeggli, Zhenbin Li, Dimitri Papadimitriou, Pierre Peloso, Reshad Rahman, Markus Stenberg, Martin Stiemerling, Rene Struik, Martin Thomson, Dacheng Zhang, and participants in the NMRG research group, the ANIMA working group, and the IESG.

8. References

8.1. Normative References

[I-D.greevenbosch-appsawg-cbor-cddl]
Birkholz, H., Vigano, C., and C. Bormann, "Concise data definition language (CDDL): a notational convention to express CBOR data structures", draft-greevenbosch-appsawg-cbor-cddl-11 (work in progress), July 2017.

[I-D.ietf-anima-autonomic-control-plane]
Behringer, M., Eckert, T., and S. Bjarnason, "An Autonomic Control Plane", draft-ietf-anima-autonomic-control-plane-07 (work in progress), July 2017.

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

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <http://www.rfc-editor.org/info/rfc3629>.

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

[RFC4086] Eastlake 3rd, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, DOI 10.17487/RFC4086, June 2005, <http://www.rfc-editor.org/info/rfc4086>.

[RFC7049] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", RFC 7049, DOI 10.17487/RFC7049, October 2013, <http://www.rfc-editor.org/info/rfc7049>.

[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, <http://www.rfc-editor.org/info/rfc7217>.

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

8.2. Informative References

[I-D.carpenter-anima-asa-guidelines]
Carpenter, B. and S. Jiang, "Guidelines for Autonomic Service Agents", draft-carpenter-anima-asa-guidelines-02 (work in progress), July 2017.

[I-D.chaparadza-intarea-igcp]
Behringer, M., Chaparadza, R., Petre, R., Li, X., and H. Mahkonen, "IP based Generic Control Protocol (IGCP)", draft-chaparadza-intarea-igcp-00 (work in progress), July 2011.

[I-D.ietf-anima-bootstrapping-keyinfra]
Pritikin, M., Richardson, M., Behringer, M., Bjarnason, S., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructures (BRSKI)", draft-ietf-anima-bootstrapping-keyinfra-07 (work in progress), July 2017.

[I-D.ietf-anima-reference-model]
Behringer, M., Carpenter, B., Eckert, T., Ciavaglia, L., Pierre, P., Liu, B., Nobre, J., and J. Strassner, "A Reference Model for Autonomic Networking", draft-ietf-anima-reference-model-04 (work in progress), July 2017.

[I-D.ietf-anima-stable-connectivity]
Eckert, T. and M. Behringer, "Using Autonomic Control Plane for Stable Connectivity of Network OAM", draft-ietf-anima-stable-connectivity-03 (work in progress), July 2017.

[I-D.liu-anima-grasp-api]
Carpenter, B., Liu, B., Wang, W., and X. Gong, "Generic Autonomic Signaling Protocol Application Program Interface (GRASP API)", draft-liu-anima-grasp-api-04 (work in progress), June 2017.

[I-D.stenberg-anima-adncp]
Stenberg, M., "Autonomic Distributed Node Consensus Protocol", draft-stenberg-anima-adncp-00 (work in progress), March 2015.

[RFC2205] Braden, R., Ed., Zhang, L., Berson, S., Herzog, S., and S. Jamin, "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", RFC 2205, DOI 10.17487/RFC2205, September 1997, <http://www.rfc-editor.org/info/rfc2205>.

[RFC2334] Luciani, J., Armitage, G., Halpern, J., and N. Doraswamy, "Server Cache Synchronization Protocol (SCSP)", RFC 2334, DOI 10.17487/RFC2334, April 1998, <http://www.rfc-editor.org/info/rfc2334>.

[RFC2608] Guttman, E., Perkins, C., Veizades, J., and M. Day, "Service Location Protocol, Version 2", RFC 2608, DOI 10.17487/RFC2608, June 1999, <http://www.rfc-editor.org/info/rfc2608>.

[RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, DOI 10.17487/RFC2865, June 2000, <http://www.rfc-editor.org/info/rfc2865>.

[RFC3315] Droms, R., Ed., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, DOI 10.17487/RFC3315, July 2003, <http://www.rfc-editor.org/info/rfc3315>.

[RFC3416] Presuhn, R., Ed., "Version 2 of the Protocol Operations for the Simple Network Management Protocol (SNMP)", STD 62, RFC 3416, DOI 10.17487/RFC3416, December 2002, <http://www.rfc-editor.org/info/rfc3416>.

[RFC3493] Gilligan, R., Thomson, S., Bound, J., McCann, J., and W. Stevens, "Basic Socket Interface Extensions for IPv6", RFC 3493, DOI 10.17487/RFC3493, February 2003, <http://www.rfc-editor.org/info/rfc3493>.

[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, <http://www.rfc-editor.org/info/rfc4861>.

[RFC5612] Eronen, P. and D. Harrington, "Enterprise Number for Documentation Use", RFC 5612, DOI 10.17487/RFC5612, August 2009, <http://www.rfc-editor.org/info/rfc5612>.

[RFC5971] Schulzrinne, H. and R. Hancock, "GIST: General Internet Signalling Transport", RFC 5971, DOI 10.17487/RFC5971, October 2010, <http://www.rfc-editor.org/info/rfc5971>.

[RFC6206] Levis, P., Clausen, T., Hui, J., Gnawali, O., and J. Ko, "The Trickle Algorithm", RFC 6206, DOI 10.17487/RFC6206, March 2011, <http://www.rfc-editor.org/info/rfc6206>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <http://www.rfc-editor.org/info/rfc6241>.

[RFC6733] Fajardo, V., Ed., Arkko, J., Loughney, J., and G. Zorn, Ed., "Diameter Base Protocol", RFC 6733, DOI 10.17487/RFC6733, October 2012, <http://www.rfc-editor.org/info/rfc6733>.

[RFC6762] Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <http://www.rfc-editor.org/info/rfc6762>.

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

[RFC6887] Wing, D., Ed., Cheshire, S., Boucadair, M., Penno, R., and P. Selkirk, "Port Control Protocol (PCP)", RFC 6887, DOI 10.17487/RFC6887, April 2013, <http://www.rfc-editor.org/info/rfc6887>.

[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, <http://www.rfc-editor.org/info/rfc7558>.

[RFC7564] Saint-Andre, P. and M. Blanchet, "PRECIS Framework: Preparation, Enforcement, and Comparison of Internationalized Strings in Application Protocols", RFC 7564, DOI 10.17487/RFC7564, May 2015, <http://www.rfc-editor.org/info/rfc7564>.

[RFC7575] Behringer, M., Pritikin, M., Bjarnason, S., Clemm, A., Carpenter, B., Jiang, S., and L. Ciavaglia, "Autonomic Networking: Definitions and Design Goals", RFC 7575, DOI 10.17487/RFC7575, June 2015, <http://www.rfc-editor.org/info/rfc7575>.

[RFC7576] Jiang, S., Carpenter, B., and M. Behringer, "General Gap Analysis for Autonomic Networking", RFC 7576, DOI 10.17487/RFC7576, June 2015, <http://www.rfc-editor.org/info/rfc7576>.

[RFC7787] Stenberg, M. and S. Barth, "Distributed Node Consensus Protocol", RFC 7787, DOI 10.17487/RFC7787, April 2016, <http://www.rfc-editor.org/info/rfc7787>.

[RFC7788] Stenberg, M., Barth, S., and P. Pfister, "Home Networking Control Protocol", RFC 7788, DOI 10.17487/RFC7788, April 2016, <http://www.rfc-editor.org/info/rfc7788>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <http://www.rfc-editor.org/info/rfc8040>.

[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, <http://www.rfc-editor.org/info/rfc8126>.

Appendix A. Open Issues [RFC Editor: This section should be empty. Please remove]

- 68. (Placeholder)

Appendix B. Closed Issues [RFC Editor: Please remove]

- 1. UDP vs TCP: For now, this specification suggests UDP and TCP as message transport mechanisms. This is not clarified yet. UDP is good for short conversations, is necessary for multicast discovery, and generally fits the discovery and divert scenarios well. However, it will cause problems with large messages. TCP is good for stable and long sessions, with a little bit of time consumption during the session establishment stage. If messages exceed a reasonable MTU, a TCP mode will be required in any case. This question may be affected by the security discussion.

  RESOLVED by specifying UDP for short message and TCP for longer one.

- 2. DTLS or TLS vs built-in security mechanism. For now, this specification has chosen a PKI based built-in security mechanism based on asymmetric cryptography. However, (D)TLS might be chosen as security solution to avoid duplication of effort. It also allows essentially similar security for short messages over UDP and longer ones over TCP. The implementation trade-offs are different. The current approach requires expensive asymmetric cryptographic calculations for every message. (D)TLS has startup overheads but cheaper crypto per message. DTLS is less mature than TLS.

  RESOLVED by specifying external security (ACP or (D)TLS).

- The following open issues applied only if the original security model was retained:

  * 2.1. For replay protection, GRASP currently requires every participant to have an NTP-synchronized clock. Is this OK for low-end devices, and how does it work during device bootstrapping? We could take the Timestamp out of signature option, to become an independent and OPTIONAL (or RECOMMENDED) option.

  * 2.2. The Signature Option states that this option could be any place in a message. Wouldn't it be better to specify a position (such as the end)? That would be much simpler to implement.

RESOLVED by changing security model.

- 3. DoS Attack Protection needs work.

  RESOLVED by adding text.

- 4. Should we consider preferring a text-based approach to discovery (after the initial discovery needed for bootstrapping)? This could be a complementary mechanism for multicast based discovery, especially for a very large autonomic network. Centralized registration could be automatically deployed incrementally. At the very first stage, the repository could be empty; then it could be filled in by the objectives discovered by different devices (for example using Dynamic DNS Update). The more records are stored in the repository, the less the multicast-based discovery is needed. However, if we adopt such a mechanism, there would be challenges: stateful solution, and security.

  RESOLVED for now by adding optional use of DNS-SD by ASAs. Subsequently removed by editors as irrelevant to GRASP istelf.

- 5. Need to expand description of the minimum requirements for the specification of an individual discovery, synchronization or negotiation objective.

  RESOLVED for now by extra wording.

- 6. Use case and protocol walkthrough. A description of how a node starts up, performs discovery, and conducts negotiation and synchronisation for a sample use case would help readers to understand the applicability of this specification. Maybe it should be an artificial use case or maybe a simple real one, based on a conceptual API. However, the authors have not yet decided whether to have a separate document or have it in the protocol document.

  RESOLVED: recommend a separate document.

- 7. Cross-check against other ANIMA WG documents for consistency and gaps.

  RESOLVED: Satisfied by WGLC.

- 8. Consideration of ADNCP proposal.

  RESOLVED by adding optional use of DNCP for flooding-type synchronization.

- 9. Clarify how a GDNP instance knows whether it is running inside the ACP. (Sheng)

  RESOLVED by improved text.

- 10. Clarify how a non-ACP GDNP instance initiates (D)TLS. (Sheng)

  RESOLVED by improved text and declaring DTLS out of scope for this draft.

- 11. Clarify how UDP/TCP choice is made. (Sheng) [Like DNS? - Brian]

  RESOLVED by improved text.

- 12. Justify that IP address within ACP or (D)TLS environment is sufficient to prove AN identity; or explain how Device Identity Option is used. (Sheng)

  RESOLVED for now: we assume that all ASAs in a device are trusted as soon as the device is trusted, so they share credentials. In that case the Device Identity Option is useless. This needs to be reviewed later.

- 13. Emphasise that negotiation/synchronization are independent from discovery, although the rapid discovery mode includes the first step of a negotiation/synchronization. (Sheng)

  RESOLVED by improved text.

- 14. Do we need an unsolicited flooding mechanism for discovery (for discovery results that everyone needs), to reduce scaling impact of flooding discovery messages? (Toerless)

  RESOLVED: Yes, added to requirements and solution.

- 15. Do we need flag bits in Objective Options to distinguish distinguish Synchronization and Negotiation "Request" or rapid mode "Discovery" messages? (Bing)

  RESOLVED: yes, work on the API showed that these flags are essential.

- 16. (Related to issue 14). Should we revive the "unsolicited Response" for flooding synchronisation data? This has to be done carefully due to the well-known issues with flooding, but it could

be useful, e.g. for Intent distribution, where DNCP doesn't seem applicable.

RESOLVED: Yes, see #14.

- 17. Ensure that the discovery mechanism is completely proof against loops and protected against duplicate responses.

  RESOLVED: Added loop count mechanism.

- 18. Discuss the handling of multiple valid discovery responses.

  RESOLVED: Stated that the choice must be available to the ASA but GRASP implementation should pick a default.

- 19. Should we use a text-oriented format such as JSON/CBOR instead of native binary TLV format?

  RESOLVED: Yes, changed to CBOR.

- 20. Is the Divert option needed? If a discovery response provides a valid IP address or FQDN, the recipient doesn't gain any extra knowledge from the Divert. On the other hand, the presence of Divert informs the receiver that the target is off-link, which might be useful sometimes.

  RESOLVED: Decided to keep Divert option.

- 21. Rename the protocol as GRASP (GeneRic Autonomic Signaling Protocol)?

  RESOLVED: Yes, name changed.

- 22. Does discovery mechanism scale robustly as needed? Need hop limit on relaying?

  RESOLVED: Added hop limit.

- 23. Need more details on TTL for caching discovery responses.

  RESOLVED: Done.

- 24. Do we need "fast withdrawal" of discovery responses?

  RESOLVED: This doesn't seem necessary. If an ASA exits or stops supporting a given objective, peers will fail to start future sessions and will simply repeat discovery.

- 25. Does GDNP discovery meet the needs of multi-hop DNS-SD?

  RESOLVED: Decided not to consider this further as a GRASP protocol issue. GRASP objectives could embed DNS-SD formats if needed.

- 26. Add a URL type to the locator options (for security bootstrap etc.)

  RESOLVED: Done, later renamed as URI.

- 27. Security of Flood multicasts (Section 2.5.6.2).

  RESOLVED: added text.

- 28. Does ACP support secure link-local multicast?

  RESOLVED by new text in the Security Considerations.

- 29. PEN is used to distinguish vendor options. Would it be better to use a domain name? Anything unique will do.

  RESOLVED: Simplified this by removing PEN field and changing naming rules for objectives.

- 30. Does response to discovery require randomized delays to mitigate amplification attacks?

  RESOLVED: WG feedback is that it's unnecessary.

- 31. We have specified repeats for failed discovery etc. Is that sufficient to deal with sleeping nodes?

  RESOLVED: WG feedback is that it's unnecessary to say more.

- 32. We have one-to-one synchronization and flooding synchronization. Do we also need selective flooding to a subset of nodes?

  RESOLVED: This will be discussed as a protocol extension in a separate draft (draft-liu-anima-grasp-distribution).

- 33. Clarify if/when discovery needs to be repeated.

  RESOLVED: Done.

- 34. Clarify what is mandatory for running in ACP, expand discussion of security boundary when running with no ACP - might rely on the local PKI infrastructure.

RESOLVED: Done.

- 35. State that role-based authorization of ASAs is out of scope for GRASP. GRASP doesn't recognize/handle any "roles".

  RESOLVED: Done.

- 36. Reconsider CBOR definition for PEN syntax. (objective-name = text / [pen, text] ; pen = uint)

  RESOLVED: See issue 29.

- 37. Are URI locators really needed?

  RESOLVED: Yes, e.g. for security bootstrap discovery, but added note that addresses are the normal case (same for FQDN locators).

- 38. Is Session ID sufficient to identify relayed responses? Isn't the originator's address needed too?

  RESOLVED: Yes, this is needed for multicast messages and their responses.

- 39. Clarify that a node will contain one GRASP instance supporting multiple ASAs.

  RESOLVED: Done.

- 40. Add a "reason" code to the DECLINE option?

  RESOLVED: Done.

- 41. What happens if an ASA cannot conveniently use one of the GRASP mechanisms? Do we (a) add a message type to GRASP, or (b) simply pass the discovery results to the ASA so that it can open its own socket?

  RESOLVED: Both would be possible, but (b) is preferred.

- 42. Do we need a feature whereby an ASA can bypass the ACP and use the data plane for efficiency/throughput? This would require discovery to return non-ACP addresses and would evade ACP security.

  RESOLVED: This is considered out of scope for GRASP, but a comment has been added in security considerations.

- 43. Rapid mode synchronization and negotiation is currently limited to a single objective for simplicity of design and implementation. A future consideration is to allow multiple objectives in rapid mode for greater efficiency.

  RESOLVED: This is considered out of scope for this version.

- 44. In requirement T9, the words that encryption "may not be required in all deployments" were removed. Is that OK?.

  RESOLVED: No objections.

- 45. Device Identity Option is unused. Can we remove it completely?.

  RESOLVED: No objections. Done.

- 46. The 'initiator' field in DISCOVER, RESPONSE and FLOOD messages is intended to assist in loop prevention. However, we also have the loop count for that. Also, if we create a new Session ID each time a DISCOVER or FLOOD is relayed, that ID can be disambiguated by recipients. It would be simpler to remove the initiator from the messages, making parsing more uniform. Is that OK?

  RESOLVED: Yes. Done.

- 47. REQUEST is a dual purpose message (request negotiation or request synchronization). Would it be better to split this into two different messages (and adjust various message names accordingly)?

  RESOLVED: Yes. Done.

- 48. Should the Appendix "Capability Analysis of Current Protocols" be deleted before RFC publication?

  RESOLVED: No (per WG meeting at IETF 96).

- 49. Section 2.5.1 Should say more about signaling between two autonomic networks/domains.

  RESOLVED: Description of separate GRASP instance added.

- 50. Is Rapid mode limited to on-link only? What happens if first discovery responder does not support Rapid Mode? Section 2.5.5, Section 2.5.6)

RESOLVED: Not limited to on-link. First responder wins.

- 51. Should flooded objectives have a time-to-live before they are deleted from the flood cache? And should they be tagged in the cache with their source locator?

  RESOLVED: TTL added to Flood (and Discovery Response) messages. Cached flooded objectives must be tagged with their originating ASA locator, and multiple copies must be kept if necessary.

- 52. Describe in detail what is allowed and disallowed in an insecure instance of GRASP.

  RESOLVED: Done.

- 53. Tune IANA Considerations to support early assignment request.

- 54. Is there a highly unlikely race condition if two peers simultaneously choose the same Session ID and send each other simultaneous M_REQ_NEG messages?

  RESOLVED: Yes. Enhanced text on Session ID generation, and added precaution when receiving a Request message.

- 55. Could discovery be performed over TCP?

  RESOLVED: Unicast discovery added as an option.

- 56. Change Session-ID to 32 bits?

  RESOLVED: Done.

- 57. Add M_INVALID message?

  RESOLVED: Done.

- 58. Maximum message size?

  RESOLVED by specifying default maximum message size (2048 bytes).

- 59. Add F_NEG_DRY flag to specify a "dry run" objective?.

  RESOLVED: Done.

- 60. Change M_FLOOD syntax to associate a locator with each objective?

RESOLVED: Done.

- 61. Is the SONN constrained instance really needed?

  RESOLVED: Retained but only as an option.

- 62. Is it helpful to tag descriptive text with message names (M_DISCOVER etc.)?

  RESOLVED: Yes, done in various parts of the text.

- 63. Should encryption be MUST instead of SHOULD in Section 2.5.1 and Section 2.5.1?

  RESOLVED: Yes, MUST implement in both cases.

- 64. Should more security text be moved from the main text into the Security Considerations?

  RESOLVED: No, on AD advice.

- 65. Do we need to formally restrict Unicode characters allowed in objective names?

  RESOLVED: No, but need to point to guidance from PRECIS WG.

- 66. Split requirements into separate document?

  RESOLVED: No, on AD advice.

- 67. Remove normative dependency on draft-greevenbosch-appsawg-cbor-cddl?

  RESOLVED: No, on AD advice. In worst case, fix at AUTH48.

Appendix C. Change log [RFC Editor: Please remove]

draft-ietf-anima-grasp-15, 2017-07-07:

Updates following additional IESG comments:

Security (Eric Rescorla): missing brittleness of group security concept, attack via compromised member.

TSV (Mirja Kuehlewind): clarification on the use of UDP, TCP, mandate use of TCP (or other reliable transport).

Clarified that in ACP, UDP is not used at all.

Clarified that GRASP itself needs TCP listen port (was previously written as if this was optional).

draft-ietf-anima-grasp-14, 2017-07-02:

Updates following additional IESG comments:

Updated 2.5.1 and 2.5.2 based on IESG security feedback (specify dependency against security substrate).

Strengthened requirement for reliable transport protocol.

draft-ietf-anima-grasp-13, 2017-06-06:

Updates following additional IESG comments:

Removed all mention of TLS, including SONN, since it was under-specified.

Clarified other text about trust and security model.

Banned Rapid Mode when multicast is insecure.

Explained use of M_INVALID to support extensibility

Corrected details on discovery cache TTL and discovery timeout.

Improved description of multicast UDP w.r.t. RFC8085.

Clarified when transport connections are opened or closed.

Noted that IPPROTO values come from the Protocol Numbers registry

Protocol change: Added protocol and port numbers to URI locator.

Removed inaccurate text about routing protocols

Moved Requirements section to an Appendix.

Other editorial and technical clarifications.

draft-ietf-anima-grasp-12, 2017-05-19:

Updates following IESG comments:

Clarified that GRASP runs in a single addressing realm

Improved wording about FQDN resolution, clarified that URI usage is out of scope.

Clarified description of negotiation timeout.

Noted that 'dry run' semantics are ASA-dependent

Made the ACP a normative reference

Clarified that LL multicasts are limited to GRASP interfaces

Unicast UDP moved out of scope

Editorial clarifications

draft-ietf-anima-grasp-11, 2017-03-30:

Updates following IETF 98 discussion:

Encryption changed to a MUST implement.

Pointed to guidance on UTF-8 names.

draft-ietf-anima-grasp-10, 2017-03-10:

Updates following IETF Last call:

Protocol change: Specify that an objective with no initial value should have its value field set to CBOR 'null'.

Protocol change: Specify behavior on receiving unrecognized message type.

Noted that UTF-8 names are matched byte-for-byte.

Added brief guidance for Expert Reviewer of new generic objectives.

Numerous editorial improvements and clarifications and minor text rearrangements, none intended to change the meaning.

draft-ietf-anima-grasp-09, 2016-12-15:

Protocol change: Add F_NEG_DRY flag to specify a "dry run" objective.

Protocol change: Change M_FLOOD syntax to associate a locator with each objective.

Concentrated mentions of TLS in one section, with all details out of scope.

Clarified text around constrained instances of GRASP.

Strengthened text restricting LL addresses in locator options.

Clarified description of rapid mode processsing.

Specified that cached discovery results should not be returned on the same interface where they were learned.

Shortened text in "High Level Design Choices"

Dropped the word 'kernel' to avoid confusion with o/s kernel mode.

Editorial improvements and clarifications.

draft-ietf-anima-grasp-08, 2016-10-30:

Protocol change: Added M_INVALID message.

Protocol change: Increased Session ID space to 32 bits.

Enhanced rules to avoid Session ID clashes.

Corrected and completed description of timeouts for Request messages.

Improved wording about exponential backoff and DoS.

Clarified that discovery relaying is not done by limited security instances.

Corrected and expanded explanation of port used for Discovery Response.

Noted that Discovery message could be sent unicast in special cases.

Added paragraph on extensibility.

Specified default maximum message size.

Added Appendix for sample messages.

Added short protocol overview.

Editorial fixes, including minor re-ordering for readability.

draft-ietf-anima-grasp-07, 2016-09-13:

Protocol change: Added TTL field to Flood message (issue 51).

Protocol change: Added Locator option to Flood message (issue 51).

Protocol change: Added TTL field to Discovery Response message (corrollary to issue 51).

Clarified details of rapid mode (issues 43 and 50).

Description of inter-domain GRASP instance added (issue 49).

Description of limited security GRASP instances added (issue 52).

Strengthened advice to use TCP rather than UDP.

Updated IANA considerations and text about well-known port usage (issue 53).

Amended text about ASA authorization and roles to allow for overlapping ASAs.

Added text recommending that Flood should be repeated periodically.

Editorial fixes.

draft-ietf-anima-grasp-06, 2016-06-27:

Added text on discovery cache timeouts.

Noted that ASAs that are only initiators do not need to respond to discovery message.

Added text on unexpected address changes.

Added text on robust implementation.

Clarifications and editorial fixes for numerous review comments

Added open issues for some review comments.

draft-ietf-anima-grasp-05, 2016-05-13:

Noted in requirement T1 that it should be possible to implement ASAs independently as user space programs.

Protocol change: Added protocol number and port to discovery response. Updated protocol description, CDDL and IANA considerations accordingly.

Clarified that discovery and flood multicasts are handled by the GRASP core, not directly by ASAs.

Clarified that a node may discover an objective without supporting it for synchronization or negotiation.

Added Implementation Status section.

Added reference to SCSP.

Editorial fixes.

draft-ietf-anima-grasp-04, 2016-03-11:

Protocol change: Restored initiator field in certain messages and adjusted relaying rules to provide complete loop detection.

Updated IANA Considerations.

draft-ietf-anima-grasp-03, 2016-02-24:

Protocol change: Removed initiator field from certain messages and adjusted relaying requirement to simplify loop detection. Also clarified narrative explanation of discovery relaying.

Protocol change: Split Request message into two (Request Negotiation and Request Synchronization) and updated other message names for clarity.

Protocol change: Dropped unused Device ID option.

Further clarified text on transport layer usage.

New text about multicast insecurity in Security Considerations.

Various other clarifications and editorial fixes, including moving some material to Appendix.

draft-ietf-anima-grasp-02, 2016-01-13:

Resolved numerous issues according to WG discussions.

Renumbered requirements, added D9.

Protocol change: only allow one objective in rapid mode.

Protocol change: added optional error string to DECLINE option.

Protocol change: removed statement that seemed to say that a Request not preceded by a Discovery should cause a Discovery response. That made no sense, because there is no way the initiator would know where to send the Request.

Protocol change: Removed PEN option from vendor objectives, changed naming rule accordingly.

Protocol change: Added FLOOD message to simplify coding.

Protocol change: Added SYNCH message to simplify coding.

Protocol change: Added initiator id to DISCOVER, RESPONSE and FLOOD messages. But also allowed the relay process for DISCOVER and FLOOD to regenerate a Session ID.

Protocol change: Require that discovered addresses must be global (except during bootstrap).

Protocol change: Receiver of REQUEST message must close socket if no ASA is listening for the objective.

Protocol change: Simplified Waiting message.

Protocol change: Added No Operation message.

Renamed URL locator type as URI locator type.

Updated CDDL definition.

Various other clarifications and editorial fixes.

draft-ietf-anima-grasp-01, 2015-10-09:

Updated requirements after list discussion.

Changed from TLV to CBOR format - many detailed changes, added co-author.

Tightened up loop count and timeouts for various cases.

Noted that GRASP does not provide transactional integrity.

Various other clarifications and editorial fixes.

draft-ietf-anima-grasp-00, 2015-08-14:

File name and protocol name changed following WG adoption.

Added URL locator type.

draft-carpenter-anima-gdn-protocol-04, 2015-06-21:

Tuned wording around hierarchical structure.

Changed "device" to "ASA" in many places.

Reformulated requirements to be clear that the ASA is the main customer for signaling.

Added requirement for flooding unsolicited synch, and added it to protocol spec. Recognized DNCP as alternative for flooding synch data.

Requirements clarified, expanded and rearranged following design team discussion.

Clarified that GDNP discovery must not be a prerequisite for GDNP negotiation or synchronization (resolved issue 13).

Specified flag bits for objective options (resolved issue 15).

Clarified usage of ACP vs TLS/DTLS and TCP vs UDP (resolved issues 9,10,11).

Updated DNCP description from latest DNCP draft.

Editorial improvements.

draft-carpenter-anima-gdn-protocol-03, 2015-04-20:

Removed intrinsic security, required external security

Format changes to allow DNCP co-existence

Recognized DNS-SD as alternative discovery method.

Editorial improvements

draft-carpenter-anima-gdn-protocol-02, 2015-02-19:

Tuned requirements to clarify scope,

Clarified relationship between types of objective,

Clarified that objectives may be simple values or complex data structures,

Improved description of objective options,

Added loop-avoidance mechanisms (loop count and default timeout, limitations on discovery relaying and on unsolicited responses),

Allow multiple discovery objectives in one response,

Provided for missing or multiple discovery responses,

Indicated how modes such as "dry run" should be supported,

Minor editorial and technical corrections and clarifications,

Reorganized future work list.

draft-carpenter-anima-gdn-protocol-01, restructured the logical flow of the document, updated to describe synchronization completely, add unsolicited responses, numerous corrections and clarifications, expanded future work list, 2015-01-06.

draft-carpenter-anima-gdn-protocol-00, combination of draft-jiang-config-negotiation-ps-03 and draft-jiang-config-negotiation-protocol-02, 2014-10-08.

Appendix D. Example Message Formats

For readers unfamiliar with CBOR, this appendix shows a number of example GRASP messages conforming to the CDDL syntax given in Section 5. Each message is shown three times in the following formats:

1. CBOR diagnostic notation.

2. Similar, but showing the names of the constants. (Details of the flag bit encoding are omitted.)

3. Hexadecimal version of the CBOR wire format.

Long lines are split for display purposes only.

D.1. Discovery Example

The initiator (2001:db8:f000:baaa:28cc:dc4c:9703:6781) multicasts a discovery message looking for objective EX1:

```
[1, 13948744, h'20010db8f000baaa28ccdc4c97036781', ["EX1", 5, 2, 0]]
[M_DISCOVERY, 13948744, h'20010db8f000baaa28ccdc4c97036781',
              ["EX1", F_SYNCH_bits, 2, 0]]
h'84011a00d4d7485020010db8f000baaa28ccdc4c9703678184634558310502 00'
```

A peer (2001:0db8:f000:baaa:f000:baaa:f000:baaa) responds with a locator:

```
[2, 13948744, h'20010db8f000baaa28ccdc4c97036781', 60000,
              [103, h'20010db8f000baaaf000baaaf000baaa', 6, 49443]]
[M_RESPONSE, 13948744, h'20010db8f000baaa28ccdc4c97036781', 60000,
              [O_IPv6_LOCATOR, h'20010db8f000baaaf000baaaf000baaa',
               IPPROTO_TCP, 49443]]
h'85021a00d4d7485020010db8f000baaa28ccdc4c9703678119ea6084186750
  20010db8f000baaaf000baaaf000baaa0619c123'
```

D.2. Flood Example

The initiator multicasts a flood message. The single objective has a null locator. There is no response:

```
[9, 3504974, h'20010db8f000baaa28ccdc4c97036781', 10000,
             [["EX1", 5, 2, ["Example 1 value=", 100]],[] ] ]
[M_FLOOD, 3504974, h'20010db8f000baaa28ccdc4c97036781', 10000,
             [["EX1", F_SYNCH_bits, 2, ["Example 1 value=", 100]],[] ] ]
h'86091a00357b4e5020010db8f000baaa28ccdc4c97036781192710
  828463455831050282704578616d706c6520312076616c75653d186480'
```

D.3. Synchronization Example

Following successful discovery of objective EX2, the initiator unicasts a request:

```
[4, 4038926, ["EX2", 5, 5, 0]]
[M_REQ_SYN, 4038926, ["EX2", F_SYNCH_bits, 5, 0]]
h'83041a003da10e8463455832050500'
```

The peer responds with a value:

```
 [8, 4038926, ["EX2", 5, 5, ["Example 2 value=", 200]]]
 [M_SYNCH, 4038926, ["EX2", F_SYNCH_bits, 5, ["Example 2 value=", 200]]]
 h'83081a003da10e8463455832050582704578616d706c6520322076616c75653d18c8'
```

D.4. Simple Negotiation Example

Following successful discovery of objective EX3, the initiator unicasts a request:

```
[3, 802813, ["EX3", 3, 6, ["NZD", 47]]]
[M_REQ_NEG, 802813, ["EX3", F_NEG_bits, 6, ["NZD", 47]]]
h'83031a000c3ffd84634558330306826 34e5a44182f'
```

The peer responds with immediate acceptance. Note that no objective is needed, because the initiator's request was accepted without change:

```
[6, 802813, [101]]
[M_END , 802813, [O_ACCEPT]]
h'83061a000c3ffd811865'
```

D.5. Complete Negotiation Example

Again the initiator unicasts a request:

```
[3, 13767778, ["EX3", 3, 6, ["NZD", 410]]]
[M_REQ_NEG, 13767778, ["EX3", F_NEG_bits, 6, ["NZD", 410]]]
h'83031a00d21462846345583303068 2634e5a4419019a'
```

The responder starts to negotiate (making an offer):

```
[5, 13767778, ["EX3", 3, 6, ["NZD", 80]]]
[M_NEGOTIATE, 13767778, ["EX3", F_NEG_bits, 6, ["NZD", 80]]]
h'83051a00d21462846345583303068 2634e5a441850'
```

The initiator continues to negotiate (reducing its request, and note that the loop count is decremented):

```
[5, 13767778, ["EX3", 3, 5, ["NZD", 307]]]
[M_NEGOTIATE, 13767778, ["EX3", F_NEG_bits, 5, ["NZD", 307]]]
h'83051a00d21462846345583303058 2634e5a44190133'
```

The responder asks for more time:

```
[7, 13767778, 34965]
[M_WAIT, 13767778, 34965]
h'83071a00d21462198895'
```

The responder continues to negotiate (increasing its offer):

```
[5, 13767778, ["EX3", 3, 4, ["NZD", 120]]]
[M_NEGOTIATE, 13767778, ["EX3", F_NEG_bits, 4, ["NZD", 120]]]
h'83051a00d21462846345583303048263 4e5a441878'
```

The initiator continues to negotiate (reducing its request):

```
[5, 13767778, ["EX3", 3, 3, ["NZD", 246]]]
[M_NEGOTIATE, 13767778, ["EX3", F_NEG_bits, 3, ["NZD", 246]]]
h'83051a00d21462846345583303038263 4e5a4418f6'
```

The responder refuses to negotiate further:

```
[6, 13767778, [102, "Insufficient funds"]]
[M_END , 13767778, [O_DECLINE, "Insufficient funds"]]
h'83061a00d2146282186672496e7375666696369656e742066756e6473'
```

This negotiation has failed. If either side had sent [M_END, 13767778, [O_ACCEPT]] it would have succeeded, converging on the objective value in the preceding M_NEGOTIATE. Note that apart from the initial M_REQ_NEG, the process is symmetrical.

Appendix E. Requirement Analysis of Discovery, Synchronization and Negotiation

This section discusses the requirements for discovery, negotiation and synchronization capabilities. The primary user of the protocol is an autonomic service agent (ASA), so the requirements are mainly expressed as the features needed by an ASA. A single physical device might contain several ASAs, and a single ASA might manage several technical objectives. If a technical objective is managed by several ASAs, any necessary coordination is outside the scope of the GRASP signaling protocol. Furthermore, requirements for ASAs themselves, such as the processing of Intent [RFC7575], are out of scope for the present document.

E.1. Requirements for Discovery

D1. ASAs may be designed to manage any type of configurable device or software, as required in Appendix E.2. A basic requirement is therefore that the protocol can represent and discover any kind of technical objective (as defined in Section 2.1) among arbitrary subsets of participating nodes.

In an autonomic network we must assume that when a device starts up it has no information about any peer devices, the network structure, or what specific role it must play. The ASA(s) inside the device are in the same situation. In some cases, when a new application session starts up within a device, the device or ASA may again lack

information about relevant peers. For example, it might be necessary to set up resources on multiple other devices, coordinated and matched to each other so that there is no wasted resource. Security settings might also need updating to allow for the new device or user. The relevant peers may be different for different technical objectives. Therefore discovery needs to be repeated as often as necessary to find peers capable of acting as counterparts for each objective that a discovery initiator needs to handle. From this background we derive the next three requirements:

D2. When an ASA first starts up, it may have no knowledge of the specific network to which it is attached. Therefore the discovery process must be able to support any network scenario, assuming only that the device concerned is bootstrapped from factory condition.

D3. When an ASA starts up, it must require no configured location information about any peers in order to discover them.

D4. If an ASA supports multiple technical objectives, relevant peers may be different for different discovery objectives, so discovery needs to be performed separately to find counterparts for each objective. Thus, there must be a mechanism by which an ASA can separately discover peer ASAs for each of the technical objectives that it needs to manage, whenever necessary.

D5. Following discovery, an ASA will normally perform negotiation or synchronization for the corresponding objectives. The design should allow for this by conveniently linking discovery to negotiation and synchronization. It may provide an optional mechanism to combine discovery and negotiation/synchronization in a single protocol exchange.

D6. Some objectives may only be significant on the local link, but others may be significant across the routed network and require off-link operations. Thus, the relevant peers might be immediate neighbors on the same layer 2 link, or they might be more distant and only accessible via layer 3. The mechanism must therefore provide both on-link and off-link discovery of ASAs supporting specific technical objectives.

D7. The discovery process should be flexible enough to allow for special cases, such as the following:

- During initialization, a device must be able to establish mutual trust with autonomic nodes elsewhere in the network and participate in an authentication mechanism. Although this will inevitably start with a discovery action, it is a special case precisely because trust is not yet established. This topic is the

subject of [I-D.ietf-anima-bootstrapping-keyinfra]. We require that once trust has been established for a device, all ASAs within the device inherit the device's credentials and are also trusted. This does not preclude the device having multiple credentials.

- Depending on the type of network involved, discovery of other central functions might be needed, such as the Network Operations Center (NOC) [I-D.ietf-anima-stable-connectivity]. The protocol must be capable of supporting such discovery during initialization, as well as discovery during ongoing operation.

D8. The discovery process must not generate excessive traffic and must take account of sleeping nodes.

D9. There must be a mechanism for handling stale discovery results.

E.2. Requirements for Synchronization and Negotiation Capability

Autonomic networks need to be able to manage many different types of parameter and consider many dimensions, such as latency, load, unused or limited resources, conflicting resource requests, security settings, power saving, load balancing, etc. Status information and resource metrics need to be shared between nodes for dynamic adjustment of resources and for monitoring purposes. While this might be achieved by existing protocols when they are available, the new protocol needs to be able to support parameter exchange, including mutual synchronization, even when no negotiation as such is required. In general, these parameters do not apply to all participating nodes, but only to a subset.

SN1. A basic requirement for the protocol is therefore the ability to represent, discover, synchronize and negotiate almost any kind of network parameter among selected subsets of participating nodes.

SN2. Negotiation is an iterative request/response process that must be guaranteed to terminate (with success or failure). While tie-breaking rules must be defined specifically for each use case, the protocol should have some general mechanisms in support of loop and deadlock prevention, such as hop count limits or timeouts.

SN3. Synchronization must be possible for groups of nodes ranging from small to very large.

SN4. To avoid "reinventing the wheel", the protocol should be able to encapsulate the data formats used by existing configuration protocols (such as NETCONF/YANG) in cases where that is convenient.

SN5. Human intervention in complex situations is costly and error-prone. Therefore, synchronization or negotiation of parameters without human intervention is desirable whenever the coordination of multiple devices can improve overall network performance. It follows that the protocol's resource requirements must be small enough to fit in any device that would otherwise need human intervention. The issue of running in constrained nodes is discussed in [I-D.ietf-anima-reference-model].

SN6. Human intervention in large networks is often replaced by use of a top-down network management system (NMS). It therefore follows that the protocol, as part of the Autonomic Networking Infrastructure, should be capable of running in any device that would otherwise be managed by an NMS, and that it can co-exist with an NMS, and with protocols such as SNMP and NETCONF.

SN7. Specific autonomic features are expected to be implemented by individual ASAs, but the protocol must be general enough to allow them. Some examples follow:

- Dependencies and conflicts: In order to decide upon a configuration for a given device, the device may need information from neighbors. This can be established through the negotiation procedure, or through synchronization if that is sufficient. However, a given item in a neighbor may depend on other information from its own neighbors, which may need another negotiation or synchronization procedure to obtain or decide. Therefore, there are potential dependencies and conflicts among negotiation or synchronization procedures. Resolving dependencies and conflicts is a matter for the individual ASAs involved. To allow this, there need to be clear boundaries and convergence mechanisms for negotiations. Also some mechanisms are needed to avoid loop dependencies or uncontrolled growth in a tree of dependencies. It is the ASA designer's responsibility to avoid or detect looping dependencies or excessive growth of dependency trees. The protocol's role is limited to bilateral signaling between ASAs, and the avoidance of loops during bilateral signaling.

- Recovery from faults and identification of faulty devices should be as automatic as possible. The protocol's role is limited to discovery, synchronization and negotiation. These processes can occur at any time, and an ASA may need to repeat any of these steps when the ASA detects an event such as a negotiation counterpart failing.

- Since a major goal is to minimize human intervention, it is necessary that the network can in effect "think ahead" before

changing its parameters. One aspect of this is an ASA that relies on a knowledge base to predict network behavior. This is out of scope for the signaling protocol. However, another aspect is forecasting the effect of a change by a "dry run" negotiation before actually installing the change. Signaling a dry run is therefore a desirable feature of the protocol.

Note that management logging, monitoring, alerts and tools for intervention are required. However, these can only be features of individual ASAs, not of the protocol itself. Another document [I-D.ietf-anima-stable-connectivity] discusses how such agents may be linked into conventional OAM systems via an Autonomic Control Plane [I-D.ietf-anima-autonomic-control-plane].

SN8. The protocol will be able to deal with a wide variety of technical objectives, covering any type of network parameter. Therefore the protocol will need a flexible and easily extensible format for describing objectives. At a later stage it may be desirable to adopt an explicit information model. One consideration is whether to adopt an existing information model or to design a new one.

E.3. Specific Technical Requirements

T1. It should be convenient for ASA designers to define new technical objectives and for programmers to express them, without excessive impact on run-time efficiency and footprint. In particular, it should be convenient for ASAs to be implemented independently of each other as user space programs rather than as kernel code, where such a programming model is possible. The classes of device in which the protocol might run is discussed in [I-D.ietf-anima-reference-model].

T2. The protocol should be easily extensible in case the initially defined discovery, synchronization and negotiation mechanisms prove to be insufficient.

T3. To be a generic platform, the protocol payload format should be independent of the transport protocol or IP version. In particular, it should be able to run over IPv6 or IPv4. However, some functions, such as multicasting on a link, might need to be IP version dependent. By default, IPv6 should be preferred.

T4. The protocol must be able to access off-link counterparts via routable addresses, i.e., must not be restricted to link-local operation.

T5. It must also be possible for an external discovery mechanism to be used, if appropriate for a given technical objective. In other words, GRASP discovery must not be a prerequisite for GRASP negotiation or synchronization.

T6. The protocol must be capable of distinguishing multiple simultaneous operations with one or more peers, especially when wait states occur.

T7. Intent: Although the distribution of Intent is out of scope for this document, the protocol must not by design exclude its use for Intent distribution.

T8. Management monitoring, alerts and intervention: Devices should be able to report to a monitoring system. Some events must be able to generate operator alerts and some provision for emergency intervention must be possible (e.g. to freeze synchronization or negotiation in a mis-behaving device). These features might not use the signaling protocol itself, but its design should not exclude such use.

T9. Because this protocol may directly cause changes to device configurations and have significant impacts on a running network, all protocol exchanges need to be fully secured against forged messages and man-in-the middle attacks, and secured as much as reasonably possible against denial of service attacks. There must also be an encryption mechanism to resist unwanted monitoring. However, it is not required that the protocol itself provides these security features; it may depend on an existing secure environment.

Appendix F. Capability Analysis of Current Protocols

This appendix discusses various existing protocols with properties related to the requirements described in Appendix E. The purpose is to evaluate whether any existing protocol, or a simple combination of existing protocols, can meet those requirements.

Numerous protocols include some form of discovery, but these all appear to be very specific in their applicability. Service Location Protocol (SLP) [RFC2608] provides service discovery for managed networks, but requires configuration of its own servers. DNS-SD [RFC6763] combined with mDNS [RFC6762] provides service discovery for small networks with a single link layer. [RFC7558] aims to extend this to larger autonomous networks but this is not yet standardized. However, both SLP and DNS-SD appear to target primarily application layer services, not the layer 2 and 3 objectives relevant to basic network configuration. Both SLP and DNS-SD are text-based protocols.

Simple Network Management Protocol (SNMP) [RFC3416] uses a command/response model not well suited for peer negotiation. Network Configuration Protocol (NETCONF) [RFC6241] uses an RPC model that does allow positive or negative responses from the target system, but this is still not adequate for negotiation.

There are various existing protocols that have elementary negotiation abilities, such as Dynamic Host Configuration Protocol for IPv6 (DHCPv6) [RFC3315], Neighbor Discovery (ND) [RFC4861], Port Control Protocol (PCP) [RFC6887], Remote Authentication Dial In User Service (RADIUS) [RFC2865], Diameter [RFC6733], etc. Most of them are configuration or management protocols. However, they either provide only a simple request/response model in a master/slave context or very limited negotiation abilities.

There are some signaling protocols with an element of negotiation. For example Resource ReSerVation Protocol (RSVP) [RFC2205] was designed for negotiating quality of service parameters along the path of a unicast or multicast flow. RSVP is a very specialised protocol aimed at end-to-end flows. A more generic design is General Internet Signalling Transport (GIST) [RFC5971], but it is complex, tries to solve many problems, and is also aimed at per-flow signaling across many hops rather than at device-to-device signaling. However, we cannot completely exclude extended RSVP or GIST as a synchronization and negotiation protocol. They do not appear to be directly useable for peer discovery.

RESTCONF [RFC8040] is a protocol intended to convey NETCONF information expressed in the YANG language via HTTP, including the ability to transit HTML intermediaries. While this is a powerful approach in the context of centralised configuration of a complex network, it is not well adapted to efficient interactive negotiation between peer devices, especially simple ones that might not include YANG processing already.

The Distributed Node Consensus Protocol (DNCP) [RFC7787] is defined as a generic form of state synchronization protocol, with a proposed usage profile being the Home Networking Control Protocol (HNCP) [RFC7788] for configuring Homenet routers. A specific application of DNCP for autonomic networking was proposed in [I-D.stenberg-anima-adncp].

DNCP "is designed to provide a way for each participating node to publish a set of TLV (Type-Length-Value) tuples, and to provide a shared and common view about the data published... DNCP is most suitable for data that changes only infrequently... If constant rapid state changes are needed, the preferable choice is to use an additional point-to-point channel..."

Specific features of DNCP include:

- Every participating node has a unique node identifier.
- DNCP messages are encoded as a sequence of TLV objects, sent over unicast UDP or TCP, with or without (D)TLS security.
- Multicast is used only for discovery of DNCP neighbors when lower security is acceptable.
- Synchronization of state is maintained by a flooding process using the Trickle algorithm. There is no bilateral synchronization or negotiation capability.
- The HNCP profile of DNCP is designed to operate between directly connected neighbors on a shared link using UDP and link-local IPv6 addresses.

DNCP does not meet the needs of a general negotiation protocol, because it is designed specifically for flooding synchronization. Also, in its HNCP profile it is limited to link-local messages and to IPv6. However, at the minimum it is a very interesting test case for this style of interaction between devices without needing a central authority, and it is a proven method of network-wide state synchronization by flooding.

The Server Cache Synchronization Protocol (SCSP) [RFC2334] also describes a method for cache synchronization and cache replication among a group of nodes.

A proposal was made some years ago for an IP based Generic Control Protocol (IGCP) [I-D.chaparadza-intarea-igcp]. This was aimed at information exchange and negotiation but not directly at peer discovery. However, it has many points in common with the present work.

None of the above solutions appears to completely meet the needs of generic discovery, state synchronization and negotiation in a single solution. Many of the protocols assume that they are working in a traditional top-down or north-south scenario, rather than a fluid peer-to-peer scenario. Most of them are specialized in one way or another. As a result, we have not identified a combination of existing protocols that meets the requirements in Appendix E. Also, we have not identified a path by which one of the existing protocols could be extended to meet the requirements.

Authors' Addresses

Carsten Bormann
Universitaet Bremen TZI
Postfach 330440
D-28359 Bremen
Germany

Email: cabo@tzi.org

Brian Carpenter (editor)
Department of Computer Science
University of Auckland
PB 92019
Auckland 1142
New Zealand

Email: brian.e.carpenter@gmail.com

Bing Liu (editor)
Huawei Technologies Co., Ltd
Q14, Huawei Campus
No.156 Beiqing Road
Hai-Dian District, Beijing 100095
P.R. China

Email: leo.liubing@huawei.com

rtgwg
Internet-Draft
Intended status: Informational
Expires: May 4, 2021

Y. Li
L. Iannone
Huawei Technologies
J. He
City University of Hong Kong
L. Geng
P. Liu
China Mobile
Y. Cui
Tsinghua University

October 31, 2020

Architecture of Dynamic-Anycast in Compute First Networking (CFN-Dyncast)
draft-li-rtgwg-cfn-dyncast-architecture-00

Abstract

Compute First Networking (CFN) Dynamic Anycast refers to in-network edge computing, where a single service offered by a provider has multiple instances attached to multiple edge sites. In this scenario, flows are assigned and consistently forwarded to a specific instance through an anycast approach based on the network status as well as the status of the different instance.

This document describes an architecture for the Dynamic Anycast (Dyncast) in Compute First Networking (CFN). It provides an overview, a description of the various components, and a workflow example showing how to provide a balanced multi-edge based service in terms of both computing and networking resources through dynamic anycast in real time.

Status of This Memo

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

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

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

This Internet-Draft will expire on May 4, 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

Dynamic anycast in Compute First Networking (CFN-Dyncast) use cases and problem statements document [I-D.geng-rtgwg-cfn-dyncast-ps-usecase] shows the usage scenarios that require an edge to be dynamically selected from multiple edge sites to serve an edge computing service demand based on computing resource available at the site and network status in real time. Multiple edges provide service equivalency and service dynamism in CFN. The current network architecture in edge computing provides relatively static service dispatching, for example, to the closest edge, or to the server with the most computing resources without

considering the network status. Dynamic Anycast takes the dynamic nature of computing load as well as the network status as metrics for deciding flow's service dispatch and at the same time maintains the flow affinity in a service life cycle.

CFN-Dyncast architecture presents an anycast based service and access models. The aim is to solve the problematic aspects of existing network layer edge computing service deployment, including the unawareness of computing resource information of service, static edge selection, isolated network and computing metrics and/or slow refresh of status.

CFN-Dyncast assumes there are multiple equivalent edge instances implementing the same single service (think about the same service function instantiated on several edge nodes). A single edge node has limited computing resources attached, and different edge nodes may have different resources available such as CPU or GPU. Because multiple edge nodes are interconnected and can collaborate with each other, it is possible to balance the service load and network load in CFN. Computing resource available to serve a request is usually considered the main metric to assign a service demand to an instance of the service. However, the status of the network, in particular paths toward the instances, varies over time and may get congested, hence, becoming another key attribute to be considered. CFN-Dyncast aims at providing a layer 3 protocol framework able to dispatch the service demand to the "best" edge node in terms of both computing resources and network status, in real time and no application and/or service specific dependencies.

This document describes the a general architecture for the service notification, status update and service dispatch in CFN edge computing.

2. Definition of Terms

CFN: Compute First Networking

SID: Service ID, an anycast IP address representing a service and the clients use it to access that service. SID is independent of which service instance serves the service demand. Usually multiple service instances serve a single service.

BID: Binding ID, an address to reach a service instance for a given SID. It is usually a unicast IP. A service can be provided by multiple service instances with different BID.

CFN-Dyncast: as defined in [I-D.geng-rtgwg-cfn-dyncast-ps-usecase].

3. CFN-Dyncast Architecture Overview

Service instances can be hosted on servers, virtual machines, access routers or gateway in edge data center. The CFN node is the glue allowing CFN-Dyncast network to provide the capability to exchange the information about the computing resource information of service instances attached to it, but also to forward flows consistently toward such instances.

Figure 1 shows the architecture of CFN-Dyncast. CFN nodes are usually deployed at the edges of the operator infrastructure, where clients are connected. As such, we can consider that clients are logically connected to CFN nodes. A CFN node has the purpose to constantly direct flows coming from clients to an instance of the service the flow is supposed to go through. Service instances are initiated at different edge sites, where a CFN node is also running. A single service can have a huge number of instances running on different CFN nodes. A "Service ID" (SID) is used to uniquely identify a service, at the same time identifying the whole set of instances of that specific service, no matter where those instances are running. There can be several instances of the service running on the the same CFN node (e.g., one instance per CPU core), there can also be on several different CFN nodes (e.g., one instance per PGW-U in a 5G network). Each instance is associated to a "Binding ID" indicating where the instance is running. Hence, there is a dynamic binding between an SID (the service) and a set of BIDs (the instances of the service) and such bindings are enriched with information concerning the network state and the available resources so that at each new service request (a new flow) CFN nodes can decide which instance is the most appropriate to handle the request. This highlights the anycast part of CFN-Dyncast, since flow are routed toward one service end-point among a set of equivalent , i.e., one-to-one-out-of-many.

When a clients sends a service demand, it will be delivered to the most appropriate instance of the service attached to a CFN node. A service demand is normally the first packet of a data flow, not necessarily an explicit out of band service request. Once the CFN node has decided which instance has to serve the flow, flow affinity must be guaranteed, meaning that all packets belonging to the same flow have to go through the same service instance.

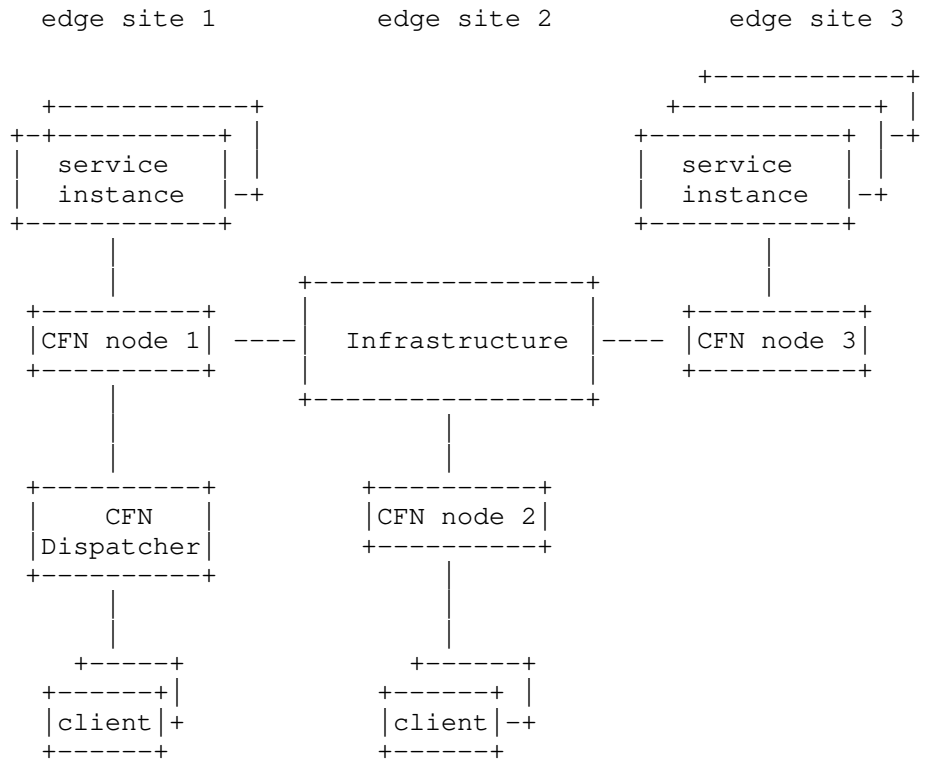


Figure 1: CFN-Dyncast Architecture

4. Architectural Components and Interactions

Figure 1 also shows that the local components of the architecture are service instance, CFN node, CFN dispatcher and client. The following subsections provide an overview of how some of these architectural components interact. The figures accompanying the examples do not show the interconnecting infrastructure to avoid making them too cluttered.

4.1. Service Identity and Bindings

As previously stated, the CFN-Dyncast architecture uses Service ID (SID) and Binding ID (BID) in order to identify services and their instances.

Service ID (SID) is an anycast service identifier (which may or may not be a routable IP address). It is used to access a specific service no matter which service instance eventually handles the

client's flow. CFN nodes must be able to know SIDs (and their bindings) in advance and must be able to identify which flow needs which service. This can be achieved in different ways, for example, use a special range or coding of anycast IP address as SID, or use DNS.

Binding ID (BID) is a unicast IP address. It is usually the interface IP address of a service instance. Mapping and binding from a SID to a BID is dynamic and depends on the computing resousrces and network state at the time the service demand is made. The CFN node must be able to guarantee flow affinity, i.e., steering the flow always toward the same instance.

Figure 2 shows an abstract example of the use of SIDs and BIDs. There are three services, namely SID1, SID2, and SID3. In particular, SID2 has two instances on different CFN nodes (CFN node 2 and CFN node 3). In this case the complete list of bindings (only in term of SID and BID, no network or resource state) are:

- SID1:BID21
- SID2:BID22,BID32
- SID3:BID33

SID: Service ID
BID: Binding ID

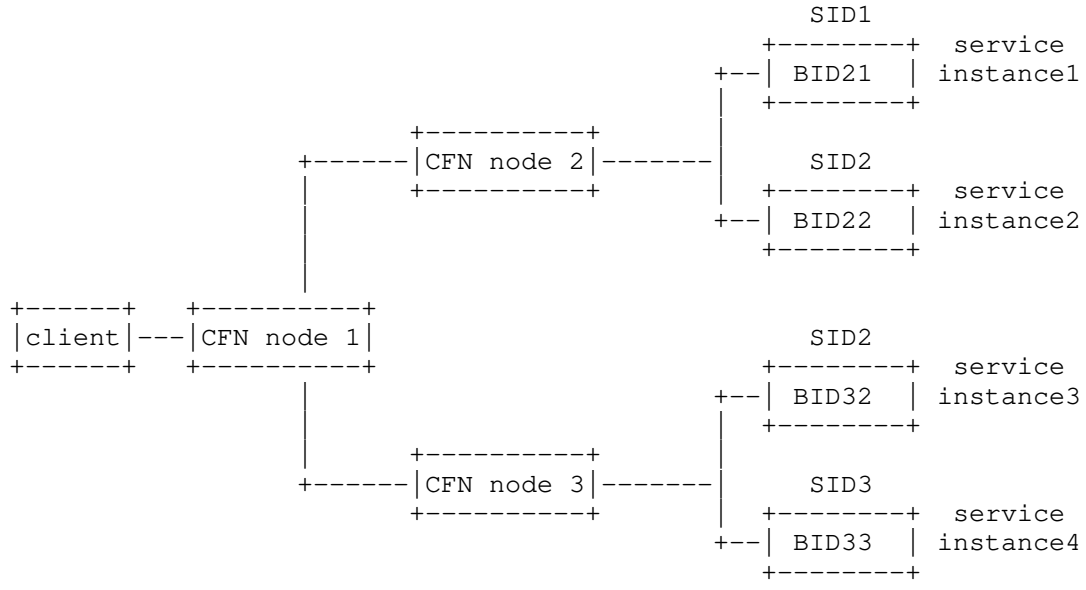


Figure 2: CFN-Dyncast Architectural Concept Example

4.2. Service Notification between Instances and CFN node

CFN-Dyncast service side is responsible to notify its attaching CFN node about the mapping information of SID and BID when a new service is instantiated, terminated, or its metrics (e.g., load) change, as shown in Figure 3.

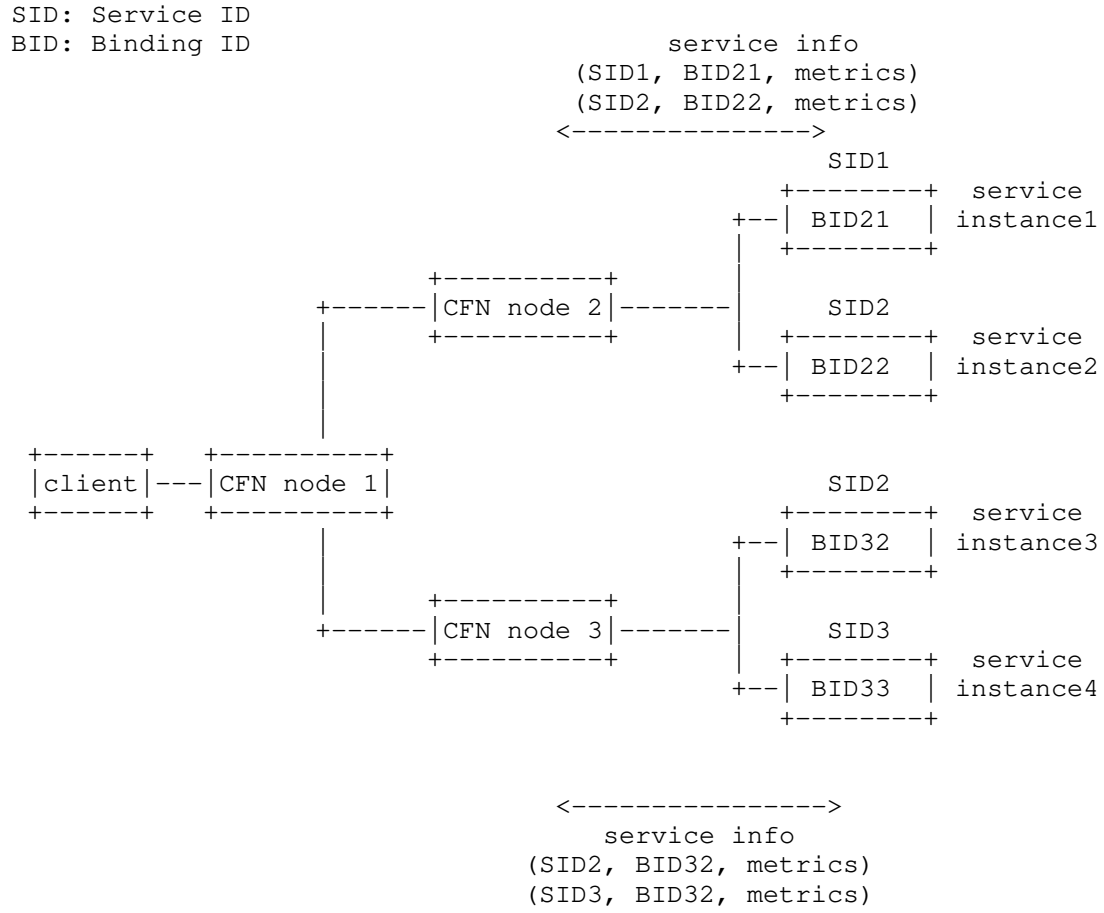


Figure 3: CFN-Dyncast Service Notification

Computing resource information of service instances is key information in CFN-Dyncast. Some of them are relatively static like CPU/GPU capacity, and some are very dynamic, for example, CPU/GPU utilization, number of sessions associated, number of queuing requests. The service side has to notify and refresh this information to its attaching CFN node. Various ways can be used, for instance via protocol or via an API of the management system. Conceptually, a CFN node keeps track of the SIDs and computing metrics of all service instances attached to it in real-time.

4.3. CFN Dyncast Control Plane

CFN Dyncast needs a control plane allowing to share information about resources and costs. Through the control plane, CFN nodes share and update among themselves the service information and the associated computing metrics for the service instances attached to it. As a network node, CFN node also monitors the network state to other CFN nodes. In this way, each CFN node is able to aggregate the information and create a complete vision of the resources avaible and the cost to reach them. For instance, for the scenario in Figure 3, the different CFN nodes will learn that there exists two instances of SID2, each of which has a certain computational capacity expressed in the metrics. Different mechanisms can be used in updating the status, for instance, BGP [RFC4760], IGP or controller based mechanism.

An important question CFN Dyncast raises is on the different ways to represent the computing metrics. A single digitalized value calculated from weighted attributes like CPU/GPU consumption and/or number of sessions associated may be the easiest. However, it may not accurately reflect the computing resources of interest. Multi-dimensional variables may give finer information, however the structure and the algorithmic processing should be sufficiently general to accommodate different type of services (i.e., metrics).

A second important issue is related to the system stability and signaling overhead. As computing metrics may change very frequently, when and how frequent such information should be exchanged among CFN nodes should be determined. A spectrum of approaches can be employed, interval based update, threshold update, policy based update, etc.

4.4. Service Demand Dispatching

Assuming that the set of metric are well defined and that the update rate is tailored so to have a stable system, the CFN Dyncast data plane has the task to dispatch flows to the "best" service instance. When a new flow comes to a CFN ingress, CFN ingress node selects the most appropriate CFN egress in terms of the network status and the computing resources of the attached service instances and guarantees flow affinity for the flow from now on.

Flow affinity is one of the critical features that CFN-Dyncast should support. The flow affinity means the packets from the same flow for a service should always be sent to the same CFN egress to be processed by the same service instance.

At the time that the most appropriate CFN egress and service instance is determined when a new flow comes, a flow binding table should save this flow binding information which may include flow identifier, selected CFN node, affinity timeout value, etc. The subsequent packets of the flow are forwarded based on the table. Figure 4 shows an example of what a flow binding table at CFN ingress node can look like.

Flow Identifier					CFN egress	timeout
src_IP	dst_IP	src_port	dst_port	proto		
X	SID2	-	8888	tcp	CFN node 2	xxx
Y	SID2	-	8888	tcp	CFN node 3	xxx

Figure 4: Example of flow binding table

A flow entry in the flow binding table can be identified using the classic 5-tuple value. However, it is worth noting that different services may have different granularity of flow identification. For instance, an RTP video streaming may use different port numbers for video and audio, and it may be identified as two flows if 5-tuple flow identifier is used. However they certainly should be treated as the same flow. Therefore 3-tuple based flow identifier is more suitable for this case. Hence, it is desired to provide certain level of flexibility in identifying flows in order to apply flow affinity.

Flow affinity attributes information can be configured per service in advance. For each service, the information can include the flow identifier type, affinity timeout value, etc. The flow identifier type can indicate what are the values, for instance, 5-tuple, 3-tuple or anything else that can be used as the flow identifier. Because we deal with single services the matching rules have to be disjoint, meaning that two different services need not have non-overlapping matching flow set.

4.5. CFN Dispatcher

When a CFN node maintains the flow binding table, the memory consumed is determined by the number of flows that CFN ingress node handles. The ingress node can be an edge data center gateway, hence it may cover hundreds of thousands of users and each user may have tens of flows. The memory space consumption on binding table at the CFN

ingress node can be a concern. To alleviate it, a functional entity called CFN Dispatcher can help.

CFN Dispatcher is deployed closer to the clients and it normally handles the flows for a limited number of clients. In this case, the memory space required by the binding table will be much smaller. CFN dispatcher is a client side located entity which directs traffic to an CFN egress node. It is not a CFN node itself, that is to say, it does not participate in the status update about network and computing metrics among CFN nodes. CFN dispatcher does not determine the best CFN egress to forward packets for a new flow by itself. It has to learn such information from a CFN node and maintains it to ensure the flow affinity for the subsequent packets. In this way, the CFN node simply selects the most appropriate egress for the new flows and informs CFN dispatcher in explicit or implicit way. It is relieved from flow binding table maintenance.

Figure 5 shows the interaction between an CFN Dispatcher and a CFN node. After CFN node makes the service demand dispatch, it informs the CFN dispatcher about the selected CFN egress node for the flow. Then CFN dispatcher maintains the flow binding table to ensure the flow affinity. Message exchange between the CFN dispatcher and its corresponding CFN node needs to be defined. The CFN dispatcher can simply forward the first packet of a flow to the CFN node, who takes the decision of which instance to use and pushes this information in the flow binding table of the CFN dispatcher. However, in case of failures, e.g., CFN egress not reachable anymore, further interaction is needed between the CFN dispacther and the CFN node.

SID: Service ID
BID: Binding ID

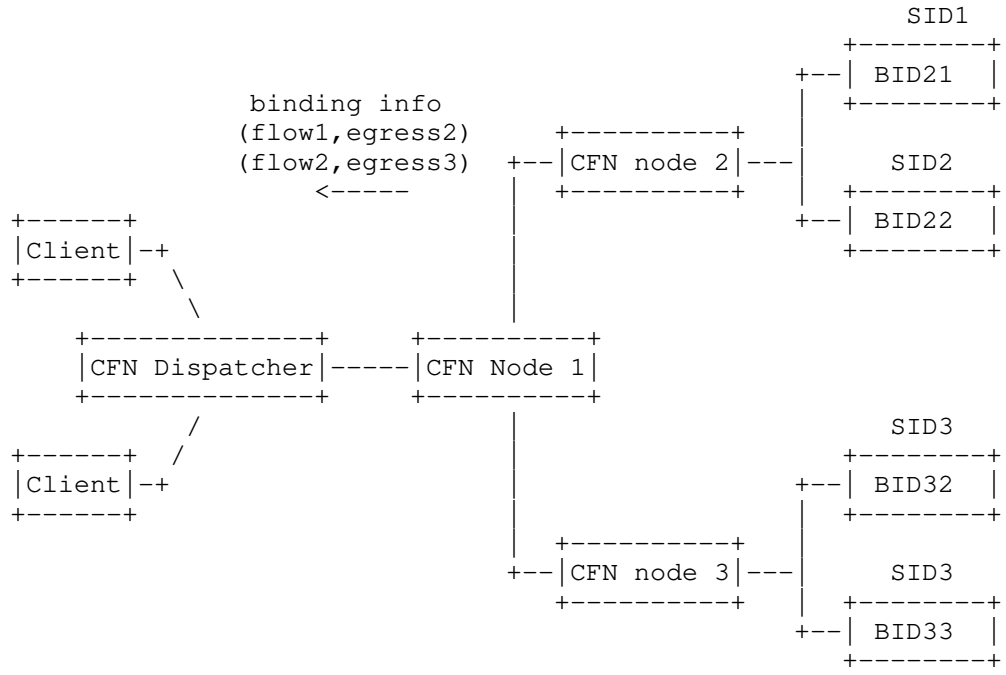


Figure 5: Service Demand Dispatch with CFN Dispatcher

5. Summary of the key elements of CFN Dyncast Architecture

- CFN Control Plane:
  - SID: CFN nodes have to made aware of existing services through the existence of the corresponding SID. It can be achieved in different ways. For example, use a special range or coding of anycast IP address as service IDs or use DNS.
  - BID bindings: SID are bound to a set of BID representing the different instances of the service. Associated to these BID there is as well a set of metrics describing the state of the instance. These bindings have to be shared among the CFN nodes so that they are aware of the different instances and their computing resource status.

    * Network state: CFN nodes have to be able to share network status so to have an idea on the impact of the dispatching decision in terms of link congestion.

    * Metric and network status updates need to be sufficiently sparse so to limit the signaling overhead and keep the system stable, but also sufficiently regular so to make the system reactive to sudden traffic fluctuations.

  o CFN Data Plane:

    * In case of a new flow: CFN ingress node selects the most appropriate CFN egress in terms of the network status and the computing resource of the service instance attached to the egresses.

    * Flow affinity: CFN ingress nodes make sure the subsequent packets of an existing flow are always delivered to the same CFN egress node so that they can be served by the same service instance.

6. Conclusion (and call for contributions)

This document introduces an architecture for CFN Dyncast, enabling the service demand request to be sent to an optimal edge to improve the overall system load balancing. It can dynamically adapt to the computing resources consumption and network status change and avoid overloading single edges. CFN-Dyncast is a network based architecture that supports a large number of edges and is independent of the applications or services hosted on the edge.

This present document is a strawman for defining CFN-Dyncast architecure.

More discussions on control plane and data plane approach are welcome.

7. Security Considerations

TBD

8. IANA Considerations

No IANA action is required so far.

9. Informative References

[RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, DOI 10.17487/RFC4760, January 2007, <https://www.rfc-editor.org/info/rfc4760>.

[I-D.geng-rtgwg-cfn-dyncast-ps-usecase] Geng, L., Liu, P., and P. Willis, "Dynamic-Anycast in Compute First Networking (CFN-Dyncast) Use Cases and Problem Statement", draft-geng-rtgwg-cfn-dyncast-ps-usecase-00 (work in progress), October 2020.

Acknowledgements

TBD

Authors' Addresses

Yizhou Li
Huawei Technologies

Email: liyizhou@huawei.com

Luigi Iannone
Huawei Technologies

Email: Luigi.iannone@huawei.com

Jianfei He
City University of Hong Kong

Email: jianfeihe2-c@my.cityu.edu.hk

Liang Geng
China Mobile

Email: gengliang@chinamobile.com

Peng Liu
China Mobile

Email: liupengyjy@chinamobile.com

Yong Cui
Tsinghua University

Email: cuiyong@tsinghua.edu.cn

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 6, 2021

Z. Li
S. Chen
Y. Gu
Huawei

November 02, 2020

Protocol Assisted Protocol (PAP)
draft-li-rtgwg-protocol-assisted-protocol-03

Abstract

For routing protocol troubleshooting, different approaches exibit merits w.r.t. different situations. They can be generally divided into two categories, the distributive way and the centralized way. A very commonly used distributive approach is to log in possiblly all related devices one by one to check massive data via CLI. Such approach provides very detailed device information, however it requires operators with high NOC (Network Operation Center) experience and suffers from low troubleshooting efficiency and high cost. The centralized approach is realized by collecting data from devices via approaches, like the streaming Telemetry or BMP(BGP Monitoring Protocol) RFC7854 [RFC7854], for the centralized server to analyze all gathered data. Such approach allows a comprehensive view fo the whole network and facilitates automated troubleshooting, but is limited by the data collection boundary set by different management domains, as well as high network bandwidth and CPU computation costs.

This document proposes a semi-distributive and semi-centralized approach for fast routing protocol troubleshooting, localizing the target device and possibly the root cause, more precisely. It defines a new protocol, called the PAP (Protocol assisted Protocol), for devices to exchange protocol related information between each other in both active and on-demand manners. It allow devices to request specific information from other devices and receive replies to the requested data. It also allows actively transmission of information without request to inform other devices to better react w.r.t. network issues.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

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

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

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

This Internet-Draft will expire on May 6, 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

A healthy control plane, providing network connectivity, is the foundation of a well-functioning network. There have been rich routing and signaling protocols designed and used for IP networks, such as IGP (ISIS,OSPF), BGP, LDP, RSVP-TE and so on. The health issues of these protocols, such as neighbor/peer disconnect/set up failure, LSP set up failure, route flapping and so on, have been devoted with ongoing efforts for diagnosing and remediation.

1.1. Motivation

The distributive protocol troubleshooting approach is typically realized through manual per-device check. It's both time- and labor-consuming, and requires NOC experience of the operators. Amongst all, localizing the target device is usually the most diffcult and time-consuming part. For example, in the case of route loop, operators first log in a random deivce that reports TTL alarms, and then check the looped route in the Forwarding Information Base (FIB) and/or the Routing Information Base (RIB). It requires device by device check, as well as manul data correlation, to pin point to the exact responsible device, since the information retrival and analysis of such distributive way is fragmented. In addition, the low efficiency and manul troubleshooting activities may further impact new network services and/or enlarge affected areas.

The centralized network OAM, by collecting network-wide data from devices, enables automatic routing protocol troubleshooting. Date collection protocols, such as SNMP (Simple Network Management Protocol) [RFC1157], NETCONF (Network Configuration Protocol) [RFC6241], and (BMP) [RFC7854], can provide various information retrival, such as network states, routing data, configurations and so on. Such centrazlized way relies on the existence of a centralized server/controller, which is not supported by some legacy networks. What's more, even with the existence of a centralized server/ controller, it can only collect the data within its own management domain, while the cross-domain data are not available due to independent managment of different ISPs. Thus, the lack of such information may lead to troubleshooting failure. In addition, centralized approaches may suffer from high network bandwidth and CPU computation consumptions.

Another way of protocol troubleshooting is utilzing the protocol itself to convey diagnosing information. For example, some reason codes are carried in the Path-Err/ResvErr messages of RSVP-TE, so that to other nodes may know the why the tunnel fails to be set up. Such approaches is semi-distributive and semi-centralized. It does not rely on the deployment of a centralized server, but still gets partial global view of the network. However, there still requires non-trivial augementation works to existing routing protocols in order to support troubleshooting. This then raises the question that whether such non-routing data is suitable to be carried in these routing protocols. The extra encapsulation, parsing and analyzing work for the non-routing data would further slow down the network convergence. Thus, it's better to separate the routing and non-routing data transmission as well as data parsing. In addition, coexisting with legacy devices may cause interop issues. Thus, relying on augumenting existing routing protocols without network-wide upgrading may not only fail to provide the truobleshooting benefit, but further affect the operation of the existing routing system. What's more, the failure of routing protocol instance would lead to the failure of diagnosing itself. All in all, it's reasonable to separate the protocol diagnosing data generation/encapsulation/transmission/parsing from the protocol itself.

This document proposes a new protocol, called the PAP (Protocol assisted Protocol), for devices to exchange protocol related information between each other. It allows both active and on-demand data exchange. Considering that massiveness of protocol/routing related data, the intuitive of designing PAP is not to exchange the comprehensive protocol/routing status between devices, but to provide very specific information required for fast troubleshooting. The

benefits of such a semi-distributive and semi-centralized approach are summarized as follows:

1. It facilitates automatic troubleshooting without requiring manul device by device check.

2. It allows individual device to have a more global view by requesting data from other devices.

3. It does not rely on the deployement of a centralized server/ controller.

4. It passes the dtata collection boundary set by different management domains by cross-domain data exchange between devices.

5. It relieves the bandwidth pressure of network-wide data collection, and the processing pressure of the centralized server.

6. It does not affect the running of existing routing protocols.

1.2. PAP Usage Use cases

PAP allows both data request/reply and data notification between devices. PAP speakers use the exchanged PAP data to help fast localize the network issues.

1.2.1. Use Case 1: BGP Route Oscillation

A BGP route oscillation can be caused by various reasons, and usually leaves network-wide impact. In order to find the root cause and take remediation actions, the first step is to localize the oscillation source. In this case, a BGP speaker can send a PAP Request Message to the next hop device of the oscillating route asking " Are you the oscillation source?". If the BGP speaker is the oscillation source, possiblly knows by running a device diagnosing system, replies with a PAP Reply Message saying that "I’m the oscillation source!" to the device who sends the PAP Request Message. If the BGP speaker is not the oscillation source, it further asks the same question with a PAP Request Message to its next hop device of the oscillating route. This request and reply process continues util the request has reached the oscillation source. The source device then sends a PAP Reply Message to tell its upstream device along the PAP request path that " I am the oscillation source!", and then "xx is the oscillation source!" information is further sent back hop by hop to the device who originates the request.

1.2.2. Use Case 2: RSVP-TE Set Up Failure

The MPLS label switch path set up, either using RSVP-TE or LDP, may fail due to various reasons. Typical troubleshooting procedures are to log in the device, and then check if the failure lies on the configuration, or path computation error, or link failure. Sometimes, it requires the check of multiple devices along the tunnel. Certain reason codes can be carried in the Path-Err/ResvErr messages of RSVP-TE, while other data are currently not supported to be transmitted to the path ingress/egress node, such as the authentication failure. Using PAP, the device, which is reponsible for the tunnel set up failure, can send the PAP Notification Message to the Ingress device, and possibly with some reason codes so that the ingress device can not only localize the target device but also the root cause.

2. Terminology

IGP: Interior Gateway Protocol

IS-IS: Intermediate System to Intermediate System

OSPF: Open Shortest Path First

BGP: Boarder Gateway Protocol

BGP-LS: Boarder Gateway Protocol-Link State

MPLS: Multi-Protocol Label Switching

RSVP-TE: Resource Reservation Protocol-Traffic Engineering

LDP: Label Distribution Protocol

BMP: BGP Monitoring Protocol

LSP: Link State Packet

IPFIX: Internet Protocol Flow Information Export

PAP: Protocol assisted Protocol

UDP: User Datagram Protocol

3. PAP Overview

3.1. PAP Encapsulation

PAP uses UDP as its transport protocol, which is connectionless. The reason that UDP is selected over TCP is because PAP is intended for on-demand communications. The PAP packet is defined as follows. This document requires the assignment of a User Port registry for the UDP Destination Port.

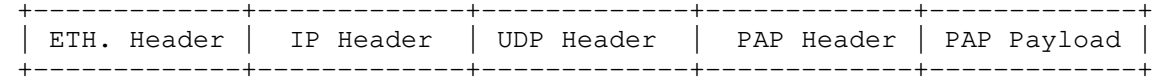


Figure 1. Encapsulation in UDP

3.2. PAP Speaker and PAP Agent

This document uses PAP speakers to refer to routing devices that communicate with each other using PAP. PAP speakers SHOULD be implemented with a supporting module (or multiple modules) to receive, parse, analyze, generate, and send PAP messages. For example, a BGP diagnosing module used for BGP related PAP message handling functions as a PAP agent. A PAP Agent is the union of multiple such modules regarding different protocols, or one module for all protocols. Such supporting module is called PAP Agent in this document. PAP Agent, standalone, SHOULD be able to provide protocol troubleshooting capability with local information. Enabling PAP exchange capability, PAP agent gains information from remote PAP speakers to improve diagnosing accuracy . The primary function of PAP is to provide a unfied tunnel for protocol diagnosing information exchange without augumenting each specific protocol.

3.3. PAP Event

A PAP Event is referred to as the a troubleshooting instance running within a PAP Agent. A PAP Agent may instantiate one or multiple PAP Events for each protocol at the same time depending on the configured troubleshooting triggering condition. For example, an PAP Event is intiated automatically when device CPU is over high, or manually with related command line input from a device operator. Once a PAP Event is generated, corresponding PAP processes are to be called on demand. Notice, the initiation of PAP Capability Negotiation does not require the existance of a PAP Event.

3.4. Summary of Operation

The communications between two PAP speakders should follow three major processes, i.e., the Capability Negotiation Process, the Request and Reply Process, and the Notification Process. This document defines 5 PAP Message types, i.e., Negotiation Message, Request Message, Reply Message, Notification Message, and ACK Message, which are used in the above PAP processes.

3.4.1. PAP Capability Negotiation Process

The purpose of the Capability Negotiation process is to inform two PAP speakers of each other's PAP capabilties. The PAP capability indicates, for which specific protocol(s), that PAP supports its/ their diagnosing information exchange. The process can be further divided into three procudures: 1) PAP Peering Relations Establish process, 2) PAP Capability Enabling Notification Process, 3) PAP Capability Disabling Notification Process. The Capability Negotiation Process is realized by the exchange of PAP Capability Negotiation Message, which is defined in Section 4.

Although PAP is connectionless, a successful PAP Peering Relations Establish Process is required to be successfully performed before any other PAP process. This process can be initiated by either the local or remote PAP speaker through sending out a PAP Capability Negotiation Message. The Negotiation Message may or may not require an ACK Message, as indicated in the Negotiation Message. A successful Peering is established if both PAP speakers have correctly received the other speaker's Capability Negotiation Message. After a successful negotiation, two PAP speakers can exchange any PAP Message on-demand. The PAP Capability Enabling Notification Process is used to inform the PAP peer its newly supported capability, which can be intiated by the PAP speaker at any moment after a PAP Peering is established with the respective PAP Peer. The PAP Capability Disabling Notification Process is used to inform the PAP peer its newly unsupported capability, which can be intiated by the PAP speaker at any moment after a PAP Peering is established with the respective PAP Peer.

3.4.2. PAP Request and Reply Process

The purpose of the PAP Request and Reply Process is to acquire information needed by a PAP speaker from other PAP speakers for a specific PAP Event. The Request and Reply Messages can be customized for different events. The process is triggered by the instantiation of a PAP Event, and starts with sending a Request Message to a target PAP peer. The target PAP peer is selected by the PAP agent regarding the current PAP Event, which is out of the scope of this document.

The remote PAP speaker, after receiving the Request Message, sends out a Reply Message to the request sender. ACK is required or not as indicated in the Message Flag.

One Request Message received at the local PAP speaker from a PAP peer may further results in a new Request Message generation regarding a third PAP speaker, if the local PAP speaker does not have the right Reply to this PAP peer. This local PAP speaker does not send Reply Message to the requesting PAP peer until it receives a new Reply Message from this third PAP speaker. So the whole process In order to avoid Request/Reply loops, a Residua Hop value is used to limit the Request/Reply rounds.

3.4.3. PAP Notification Process

The Notification Process is used by a PAP speaker voluntarily to notify other PAP speakers of certain information regarding a PAP Event. The process is triggered by the instantiation of a PAP Event, and starts with sending a Notification Message to one or multiple target PAP peer(s). The target PAP peer(s) is/are selected by the PAP agent regarding the current PAP Event, which is out of the scope of this document. The Notification Message may or may not require an ACK Message, as indicated in the Notification Message.

4. PAP Message Format

4.1. Common Header

The common header is encapsulated in all PAP messages. It is defined as follows.

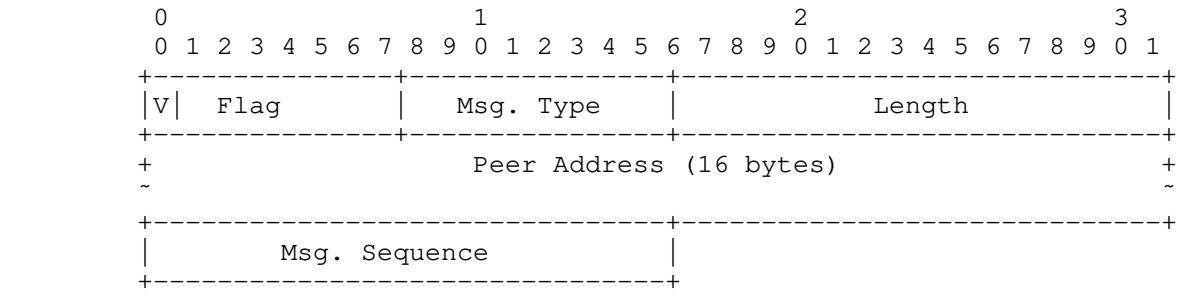


Figure 2. PAP Common Header

- Flag (1 byte): The V flag indicates that the source IP address is an IPv6 address. For IPv4 address, this is set to 0.

- Message Type (1 byte): This indicates the PAP message type.The following types are defined, and listed as follows.
  - Type = TBD1: Capability Negotiation Message. It is used for two devices to inform each other of the capabilties they support and no longer support.
  - Type = TBD2: Request Message.
  - Type = TBD3: Reply Message.
  - Type = TBD4: Notification Message.
  - Type = TBD5: ACK Message. It is used to confirm to the local device that the remote device has received a previous sent PAP message, which can be either a Negotiation Message, a Request Message, a Reply Message or an Notification Message.
- Length (2 bytes): Length of the message in bytes, including the Common Header and the following Message.
- Souece IP Address (16 bytes): It indicates the IP address who initiates the PAP message. It is 4 bytes long if an IPv4 address is carried in this field (with the 12 most significant bytes zero-filled) and 16 bytes long if an IPv6 address is carried in this field.
- Message Sequence (2 bytes): It indicates the sequence number of each PAP message.

4.1.1. Capability Negotiation Message

The Negotiation Message is used in the PAP Capability Negotiation Process. It is defined as follows.

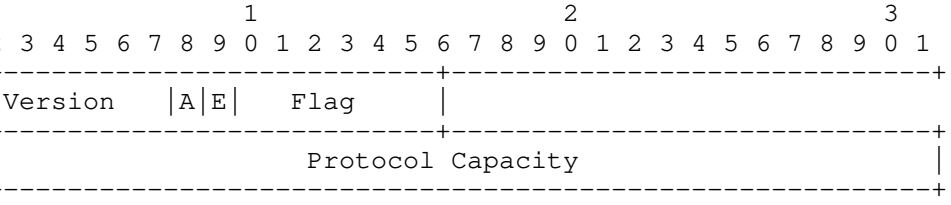


Figure 3. PAP Negotiation Message

- Version (1 byte): It indicates the PAP version. The current version is 0.

- Flags (1 bytes): Two flag bits are currently defined.
  - The A bit is used to indicate if an ACK Message from the remote PAP speaker is required for each Negotiation Message sent. If an ACK is required, then the A bit SHOULD be set to "1", and "0" otherwise.
  - The E bit is used to indicate the enabling/disabling of the capabilities that carried in the Protocol Capability field. If the local device wants to inform the remote device of enabling one or more capabilities, the E bit SHOULD be set to "1". If the local device wants to inform the remote device of disabling one or more capabilities, the E bit SHOULD be set to "0".
- Protocol Capability (4 bytes): It is 4-byte bitmap that indicates the capability of inforamtion exchange regarding various protocols. Each bit represents one protocol. The following protocol capability is defined (from the rightmost bit).
  - Bit 0: ISIS
  - Bit 1: OSPF
  - Bit 2: BGP
  - Bit 3: LDP

4.2. Request Message

The Request Message is used for the local device to request specific data regarding one specific protocol or application from the remote device. It MUST be sent after a successful Capability Negotiation Process (described in Section 5.1), and the requested protocol/ application MUST be supported by both the local and remote devices, as indicated in the Negotiation Messages exchanged between the local and remote devices. It is defined as follows.

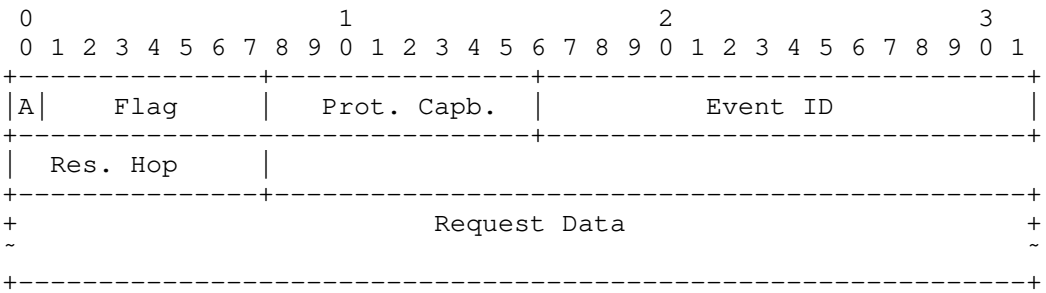


Figure 4. PAP Request Message

- Flags (1 byte): It is currently reserved. The A bit is used to indicate if an ACK Message from the remote PAP speaker is required for each Request Message sent. If an ACK is required, then the A bit SHOULD be set to "1", and "0" otherwise.

- Capability (1 byte): It represents the bit index of the protocol, which the Request Message is requesting data for.

- Event ID (2 bytes): It indicates the event number that this Request message is regarding.

- Residua hop (1 byte): it indicates the residua Request hops of the current PAP Event. It is reduced by 1 at each PAP speaker when generating a further PAP Request to a third PAP speaker.

- Request Data (Variable): Specifies information of the data that the local device is requesting. The specific format remains to be determined per each protocol, as well as each use case.

4.3. Reply Message

The Reply Message is used to carry the information that the local device requests from the remote device through the Request Message. It is defined as follows.

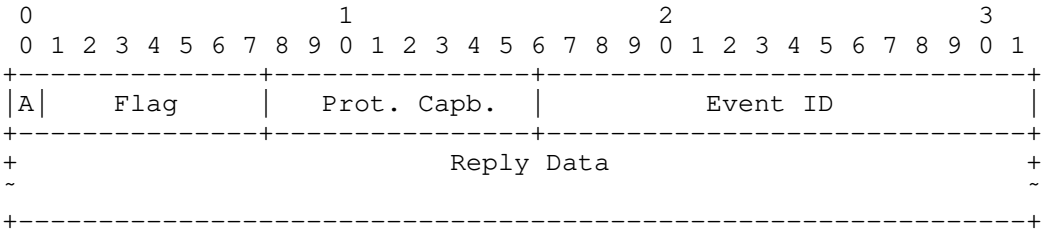


Figure 5. PAP Reply Message

- Flags (1 byte): It is currently reserved. The A bit is used to indicate if an ACK Message from the remote PAP speaker is required for each Reply Message sent. If an ACK is required, then the A bit SHOULD be set to "1", and "0" otherwise.

- Capability (1 byte): It represents the bit index of the protocol, which the Reply Message is replying data for.

- Event ID (2 bytes): It indicates the event number that this Reply message is regarding.

- Reply Data (Variable): Specifies information of the data that the local device is replying. The specific format remains to be determined per each protocol, as well as each use case.

4.4. Notification Message

The Notification Message is used to carry the information that the local device sends to the remote device.

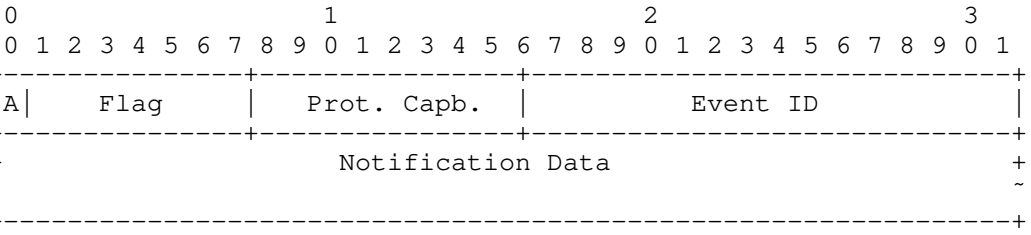


Figure 6. PAP Notification Message

- Flags (1 byte): It is currently reserved. The A bit is used to indicate if an ACK Message from the remote PAP speaker is required for each Notification Message sent. If an ACK is required, then the A bit SHOULD be set to "1", and "0" otherwise.

- Capability (1 byte): It represents the bit index of the protocol, which the Notification Message is notifying for.

- Event ID (2 bytes): It indicates the event number that this Notification Message is regarding.

- Notification Data (Variable): Specifies information of the data that the local device is notifying. The specific format remains to be determined per each protocol, as well as each use case.

-

4.5. ACK Message

The ACK Message is used to confirm that the remote device has received a PAP Message with the A bit set to "1". The ACK Message includes only the PAP Common Header. The Msg. Sequence MUST be set to the sequence number carried in the received PAP message, which requires this ACK.

5. PAP Operations

The PAP operations include the following 3 major processes, the Capability Negotiation Process, the Data Request and Reply Process, and the Data Notification Process.

5.1. Capability Negotiation Process

5.1.1. PAP Peering Relation Establish Process

A successful PAP Peering relation MUST be Established between two PAP speakers before any other PAP process.

As the first step, a Capability Negotiation Message can be initiated at any time by a PAP speaker,as long as the target PAP peer is IP reachable. It usually companies the establishment of neighboring/ peering relation between two routing devices. The "A" bit in the Negotiation Message MUST be set as 1 during the PAP Peering Establish Process, meaning ACK required. The "E" in the Negotiation Message MUST be set to 1 during this process, meaning the capabilities indicated in the Protocol Capability field are enabled by default. The Protocol Capability field SHOULD indicate all the protocol capabilities that are supported by the local PAP Agent and currently enabled. After the first Negotiation Message is sent, the local device SHUOLD wait for the ACK Message from the remote device for a certain time period before taking further actions, and if no ACK Message is received within this time frame, the local device SHOULD resend the Negotiation Message to the remote device. The waiting period can be configured locally. This send and wait process CAN be repeated for at most 3 times before receiving a ACK Message from the remote device. If after 3 times of resending the Negotiation Message, still no ACK received, then this peering establishment is treated as unsuccessful.

The next step for the local PAP speaker is to wait for the Negotiation Message from the remote PAP speaker. If no Negotiation Message is received from the remote PAP speaker within a time frame after its own Negotiation Message is sent , the local PAP speaker CAN resend the Negotiation Message. This time frame is also configured locally. This send and wait process CAN be repeated for at most 3

times before receiving a Negotiation Message from the remote PAP speaker. If after 3 times of resending the Negotiation Message, still no Negotiation Message received, then this negotiation is treated as unsuccessful. If a Negotiation Message is received and parsed correctly, an ACK MUST be sent to the remote PAP speaker.

Once an ACK Message and a Negotiation Message are received from the remote PAP speaker and correctly parsed, a PAP Peering relation is considered as successfully established. The local PAP speaker maintains locally the protocol capabilities of the remote PAP speaker, and uses them during other PAP processes.

5.1.2. PAP Capability Enabling Notification Process

Once the PAP Peering relation is set up between two PAP speakers, they become PAP peers. Thereafter, any PAP speaker supports a new protocol capability, it SHOULD call the Capability Enabling Notification Process to inform all its PAP peers.

When the local PAP speaker initates a PAP Capability Enabling Notification Process: The "A" bit in the Negotiation Message MUST be set as 1 during the PAP Capability Enabling Notification Process, meaning ACK required. The "E" in the Negotiation Message MUST be set to 1 during this process, meaning the capabilities indicated in the Protocol Capability field are enabled. The Protocol Capability field SHOULD indicate all the protocol capabilities that are supported by the local PAP Agent and currently enabled. After the Negotiation Message is sent, the local PAP speaker SHUOLD wait for the ACK Message from the PAP peer for a certain time period before taking further actions, and if no ACK Message is received within this time frame, the local device SHOULD resend the Negotiation Message to the remote device. The waiting period can be configured locally. This send and wait process CAN be repeated for at most 3 times before receiving a ACK Message from the remote device. If after 3 times of resending the Negotiation Message, still no ACK received, then this Capability Enabling Notification Process is treated as unsuccessful. This process MAY be intiated at another time thereafter. If a ACK is received, the Capability Enabling Notification Process is considered successful.

When a PAP peer initates a PAP Capability Enabling Notification Process: The local PAP speaker, after receiving the PAP Negotiation Message and correctly parsing it, sends out an ACK. This Capability Enabling Notification Process is considered successful. The local PAP speaker updates the capability status maintained accordingly.

5.1.3. PAP Capability Disabling Notification Process

Whenever a PAP speaker disables a PAP capability, it SHOULD initiate a PAP Capability Disabling Notification Process to inform all its PAP peers.

When the local PAP speaker initates a PAP Capability Disabling Notification Process: The "A" bit in the Negotiation Message MUST be set as 1 during the PAP Capability Disabling Notification Process, meaning ACK required. The "E" in the Negotiation Message MUST be set to 0 during this process, meaning the capabilities indicated in the Protocol Capability field are disabled. The Protocol Capability field SHOULD indicate all the protocol capability that is disabled. After the Negotiation Message is sent, the local PAP speaker SHUOLD wait for the ACK Message from the PAP peer for a certain time period before taking further actions, and if no ACK Message is received within this time frame, the local device SHOULD resend the Negotiation Message to the remote device. The waiting period can be configured locally. This send and wait process CAN be repeated for at most 3 times before receiving a ACK Message from the remote device. If after 3 times of resending the Negotiation Message, still no ACK received, then this Capability Disabling Notification Process is treated as unsuccessful. This process MAY be intiated at another time thereafter.

When a PAP peer initates a PAP Capability Disabling Notification Process: The local PAP speaker, after receiving the PAP Negotiation Message and correctly parsing it, sends out an ACK. This Capability Disabling Notification Process is considered successful. The local PAP speaker updates the capability status maintained accordingly.

5.2. PAP Request and Reply Process

When a local PAP Event triggers a PAP Request and Reply Process, the local PAP speaker initates a Request Message, and send to a target PAP peer as indicated by PAP Agent per this PAP Event. This local PAP speaker is called the Request and Reply Process Starter. It sets the Residua Hop as the maximum number of Request/Reply rounds (e.g., 10) it will wait in order to receive the final Reply. The Event ID and the Request are set by the local PAP Agent. The A bit of the Request Message MUST be set to "1" (i.e., ACK is required). The local device waits for the ACK Message from the remote device for a certain time period before taking further actions, and if no ACK Message is received within this time frame, the local device SHOULD resend the Request Message to the remote device. The waiting period can be configured locally. This send and wait process CAN be repeated for at most 3 times before receiving a ACK Message from the remote device. If after 3 times of resending the Request Message,

still no ACK received, then this Request and Reply Process is treated as unsuccessful. If ACK received, the local device waits for the Reply Message. If no Reply Message is received from the remote device within a time frame, the local device can resend the Request Message. This send and wait process CAN be repeated for at most 3 times before receiving a Reply Message from the remote device. If after 3 times of resending the Request Message, still no Reply Message received, then this Request and Reply Process is treated as unsuccessful. The waiting period can be configured locally, and SHOULD take into consideration of the Residua Hop value. If the Request and Reply Process Starter receives the Reply Message within the time frame, and the Event ID is matched to the local PAP Event, the PAP Request and Reply Process is considered as successful.

When a local PAP speaker receives a Request Message from its PAP peer (i.e., it is not the Pequest and Reply Process Starter), it sends back an ACK Message. With the received Request Message, a new PAP event it instantiated at the local PAP Agent. The PAP event triggers the troubleshooting analysis of the received Request Message, and then generate the Reply Message if the Reply condition is met, or generate a new Request Message when the Reply condition is not met. The Reply condition and the troubleshooting analysis of the PAP Agent is out of the scope of this document.

If the Reply condition is met, the local PAP speaker is called the Request and Reply Process Terminator. It generates the Reply Message and send the message back to the requesting PAP peer. The Event ID is set to be the same as the Event ID of the received Request Message. The Reply Data is set by the local PAP Agent per this generated event. The A bit of the Reply Message MUST be set to "1" (i.e., ACK is required). The local device waits for the ACK Message from the remote device for a certain time period before taking further actions, and if no ACK Message is received within this time frame, the local device SHOULD resend the Reply Message to the remote device. The waiting period can be configured locally. This send and wait process CAN be repeated for at most 3 times before receiving a ACK Message from the remote device. If after 3 times of resending the Request Message, still no ACK received, then this Request and Reply Process is treated as unsuccessful.

If the Reply condition is not met, the local PAP speaker is called the Request and Reply Process mid-handler. It generates a new Request Message and send the message to a third PAP speaker per indicated by the local PAP Agent per this generated event. In the new generated Request Message, the Residua Hop value by MUST be reduced by 1. The A bit of the Request Message MUST be set to "1" (i.e., ACK is required). The local device waits for the ACK Message from the remote device for a certain time period before taking

further actions, and if no ACK Message is received within this time frame, the local device SHOULD resend the Request Message to the remote device. The waiting period can be configured locally. This send and wait process CAN be repeated for at most 3 times before receiving a ACK Message from the remote device. If after 3 times of resending the Request Message, still no ACK received, then this Request and Reply Process is treated as unsuccessful. If ACK received, the local device waits for the Reply Message. If no Reply Message is received from the remote device within a time frame, the local device can resend the Request Message. This send and wait process CAN be repeated for at most 3 times before receiving a Reply Message from the remote device. If after 3 times of resending the Request Message, still no Reply Message received, then this Request and Reply Process is treated as unsuccessful. The waiting period can be configured locally, and SHOULD take into consideration of the Residua Hop value. If the local device receives the Reply Message within the time frame, it generates a new Reply Message and sends back to it requesting PAP peer. The Event ID of the new Reply Message is set to be the same as the Event ID of the received Request Message.

5.3. PAP Notification Process

When a local PAP Event triggers a PAP Notification Process, the local PAP speaker initates a Notification Message. The target PAP peer(s) is/are selected by the PAP agent regarding the current PAP Event, which is out of the scope of this document. The Notification Message may or may not require an ACK Message, as indicated in the Notification Message. If the A bit is set to 1 (meaning ACK required), the local device waits for the ACK Message from the remote device for a certain time period before taking further actions, and if no ACK Message is received within this time frame, the local device SHOULD resend the Notification Message to the remote device. The waiting period can be configured locally. This send and wait process CAN be repeated for at most 3 times before receiving a ACK Message from the remote device. If after 3 times of resending the Request Message, still no ACK received, then this Request and Reply Process is treated as unsuccessful. The waiting period can be configured locally. If ACK is received within the time frame, the Notification Process is considered to be successful. If the A bit is set to 0 (meaning no ACK required), after sending the Notification Message, the Notification Process is considered successful.

6. PAP Error Handling

When any PAP process is unsuccessful, information is recorded or not by local PAP Agent. No further action is taken.

7. Discussion

In addition to the preceding message definition and process description, the security and reliability requirements of the PAP need to be considered. There are two possible options to implement PAP.

- Option 1: PAP is developed independently as a new protocol.

- Option 2: PAP reuses the existing protocol Generic Autonomic Signaling Protocol(GRASP) [I-D.ietf-anima-grasp] .

Option1:

1. Definition of the Message Format and Interaction Process: It can be defined independently in the PAP.

2. Reliability: The transmission mode of PAP is based on UDP mainly considering that the collected information is the auxiliary information to help locate the protocol fault, and the information loss has no impact on the service. In addition, if TCP mode is adopted, the resource consumption of the device may be large, especially when there area large number of neighbors. If it is considered that PAP must ensure reliability, it can done in the application layer, such as adding the sequence number to the message.

3. Security: MD5 authentication can be introduced for PAP security.

Option2:

ANIMA GRASP is a signaling protocol used for dynamic peer discovery, status synchronization, and parameter negotiation between AS nodes or AS service agents. GRASP specifies that unicast packets must be transmitted based on TCP, and multicast packets (Discovery and Flood) must be transmitted based on UDP.

1. Message format and interaction process: PAP can reuse the defined messages and procedures of the GRASP. Messages defined in the PAP include Capability Negotiation Message, Request Message, Reply Message, and Negotiation Message. These message types are also defined in GRASP.

2. Reliability: TCP mode of GRASP can be used to ensure reliability for PAP. But there may be challenge for the equipment resources.

3. Security: Autonomic Control Plane(ACP) [I-D.ietf-anima-autonomic-control-plane] can be reused.

8. Security Considerations

TBD

9. IANA

TBD

10. Contributors

We thank Jiaqing Zhang (Huawei), Tao Du (Huawei) and Lei Li (Huawei) for their contributions.

11. Acknowledgments

12. References

[I-D.brockners-inband-oam-requirements]
Brockners, F., Bhandari, S., Dara, S., Pignataro, C., Gredler, H., Leddy, J., Youell, S., Mozes, D., Mizrahi, T., Lapukhov, P., and r. remy@barefootnetworks.com, "Requirements for In-situ OAM", draft-brockners-inband-oam-requirements-03 (work in progress), March 2017.

[I-D.ietf-anima-autonomic-control-plane]
Eckert, T., Behringer, M., and S. Bjarnason, "An Autonomic Control Plane (ACP)", draft-ietf-anima-autonomic-control-plane-30 (work in progress), October 2020.

[I-D.ietf-anima-grasp]
Bormann, C., Carpenter, B., and B. Liu, "A Generic Autonomic Signaling Protocol (GRASP)", draft-ietf-anima-grasp-15 (work in progress), July 2017.

[I-D.ietf-netconf-yang-push]
Clemm, A. and E. Voit, "Subscription to YANG Datastores", draft-ietf-netconf-yang-push-25 (work in progress), May 2019.

[I-D.song-ntf]
Song, H., Zhou, T., Li, Z., Fioccola, G., Li, Z., Martinez-Julia, P., Ciavaglia, L., and A. Wang, "Toward a Network Telemetry Framework", draft-song-ntf-02 (work in progress), July 2018.

[RFC1157] Case, J., Fedor, M., Schoffstall, M., and J. Davin, "Simple Network Management Protocol (SNMP)", RFC 1157, DOI 10.17487/RFC1157, May 1990, <https://www.rfc-editor.org/info/rfc1157>.

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

[RFC1195] Callon, R., "Use of OSI IS-IS for routing in TCP/IP and dual environments", RFC 1195, DOI 10.17487/RFC1195, December 1990, <https://www.rfc-editor.org/info/rfc1195>.

[RFC1213] McCloghrie, K. and M. Rose, "Management Information Base for Network Management of TCP/IP-based internets: MIB-II", STD 17, RFC 1213, DOI 10.17487/RFC1213, March 1991, <https://www.rfc-editor.org/info/rfc1213>.

[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>.

[RFC3209] Awduche, D., Berger, L., Gan, D., Li, T., Srinivasan, V., and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels", RFC 3209, DOI 10.17487/RFC3209, December 2001, <https://www.rfc-editor.org/info/rfc3209>.

[RFC3988] Black, B. and K. Kompella, "Maximum Transmission Unit Signalling Extensions for the Label Distribution Protocol", RFC 3988, DOI 10.17487/RFC3988, January 2005, <https://www.rfc-editor.org/info/rfc3988>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP", RFC 7752, DOI 10.17487/RFC7752, March 2016, <https://www.rfc-editor.org/info/rfc7752>.

[RFC7854] Scudder, J., Ed., Fernando, R., and S. Stuart, "BGP Monitoring Protocol (BMP)", RFC 7854, DOI 10.17487/RFC7854, June 2016, <https://www.rfc-editor.org/info/rfc7854>.

Authors' Addresses

Zhenbin Li
Huawei
156 Beiqing Rd
Beijing
China

Email: lizhenbin@huawei.com

Shuanglong Chen
Huawei
156 Beiqing Road
Beijing,100095
P.R. China

Email: chenshuanglong@huawei.com

Yunan Gu
Huawei
156 Beiqing Rd
Beijing
China

Email: guyunan@huawei.com

RTG Working Group
Internet Draft
Intended status: Informational
Expires: April 30, 2021

K. Majumdar
CommScope
U. Chunduri
L. Dunbar
Futurewei

October 31, 2020

Extension of Transport Aware Mobility in Data Network
draft-mcd-rtgwg-extension-tn-aware-mobility-00

Abstract

The existing Transport Network Aware Mobility for 5G [TN-AWARE-MOBILITY] draft specifies a framework for mapping the 5G mobile systems Slice and Service Types (SSTs) to corresponding underlying network paths in IP and Layer 2 Transport networks.The focus of that work is limited to the mobility domain and transport network characteristics till the UPF and doesn't go beyond the UPF to the Data Network.

To maintain E2E transport network characteristics the framework needs to be extended beyond UPF. This document describes a framework for extending the mobility aware transport network characteristics from the UPF through the Data Network.

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), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

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

The list of current Internet-Drafts can be accessed at
http://www.ietf.org/ietf/1id-abstracts.txt

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

This Internet-Draft will expire on April 23, 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 (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

The [TN-AWARE-MOBILITY] draft defines the transport path characteristics in backhaul, midhaul, and fronthaul segments between the radio side network functions and user plane functions (UPF). It describes how various transport network underlay routing mechanisms apply to the framework laid out including RSVP-TE, SR, and also a data plane agnostic integrated routing and TE mechanism - Preferred Path Routing (PPR) to map the network slice properties into the IP/L2 transport network.

The current [TN-AWARE-MOBILITY] draft doesn't extend the transport network characteristics from the UPF through the Data Network. If the user service termination happens in the data network, the Transport Path Network characteristics through the Data Network would be lost.

This proposed Extension of Transport Aware Mobility in Data Network extends the mobility aware transport network characteristics from the UPF through the Data Network.

The UPF can be placed on the edge of the network where it can perform entry or exit point to the Data Network. It can connect to a Provider Edge node as well and bring all the mobile connections in a distributed way to the Data Network.

The UPF can as well connect to the SD-WAN edge node or L3 aggregator device and would try to bring all the 5G mobility connections for small, medium, and large enterprises. This would be a scenario for Enterprise 5G.

The current draft proposes mechanisms on how mobility aware transport network characteristics to be mapped into SR-TE paths or Un-secure, Secure, Secure SR-TE paths based in the Data Network on different use cases scenarios.

2. Conventions used in this document

BSID - Binding SID

DC - Data Center

DN - Data Network (5G)

EMBB - enhanced Mobile Broadband (5G)

gNB - 5G NodeB

GTP-U - GPRS Tunneling Protocol - Userplane (3GPP)

MIOT - Massive IOT (5G)

PECP - Path Computation Element (PCE) Communication Protocol

SD-WAN - Software-Defined Wide Area Network

SID - Segment Identifier

SLA - Service Layer Agreement

SST - Slice and Service Types (5G)

SR - Segment Routing

SR-PCE - SR Path Computation Element

UE - User Equipment

UPF - User Plane Function (5G)

URLLC - Ultra reliable and low latency communications (5G)

3. Framework for Extension of Transport Network Aware Mobility

Architecture wise, the proposed Extension of Transport Aware Mobility in the Data Network solution focuses on the following areas:

a) The Mobility packet transition in and out from the UPF to the C-PE Node maintaining the Transport Path Characteristics.

b) On a PE node, based on the transport characteristics, use different methods of fetching SR-TE path segments from the SR-TE

Controller and map the SR-TE segments with the mobility aware transport packets.

c) On an SD-WAN CE Node, based on the transport characteristics, mapping of mobility aware transport packets to the secure and un-secure tunnel path.

Figure 1 captured under Section 4 provides the representation of a network on how UE could be connected to the UPF and C-PE nodes in the Data Network. The C-PE node represents a combined CE and PE node. In some cases, UPF would be connected to the pure PE or CE node.

4. Mobility Packet Transition to the Data Network

As the Transport Aware Mobility packets transition in and out from the UPF to the PE or C-PE (in SDWAN case) node, the Mobility Transport Characteristics need to be maintained in the Data Network. The current solution proposes a generic approach to how the mobility packet transition can happen in the Data Network maintaining the same transport characteristics. Whether the UPF would be co-located with the C-PE in the same device or sitting in a different device the approach would be the same.

The current solution proposes to create a new encapsulation header at the UPF node carrying the original UDP header along with the Inner IP to get encapsulated with the outer IP header of the outgoing C-PE node IP address.

. Format of the new Header from the UPF to the C-PE Node:
Outer IP (C-PE Node Address) + Original UDP + Inner IP (UE Packet)

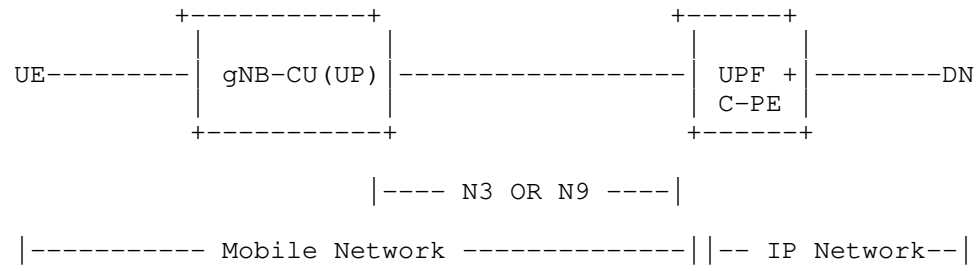
. Format of the new Header from C-PE to the UPF Node:
Outer IP (UPF Node Address) + Original UDP + Inner IP (UE Packet)

There are two possible scenarios of how UPF would be connected to the C-PE node.

In different edge networking deployment, the virtual UPF could be co-located with the C-PE node in the same device and that is captured under scenario 1. The other scenario is where UPF would be separated physically from the C-PE node over an IP network, and that is captured under Scenario 2.

In Scenario 1, the UDP source port information coming from the mobility domain can be passed to the C-PE node locally through some policy defined in the device. It doesn't require to form an IP packet with the UDP source port info to send it to the physically separated C-PE node. Figure 2 is applicable for Scenario 2, where UPF needs to forward the IP packet with the original UDP source port information to the physically separated C-PE node.

Scenario 1:



Scenario 2:

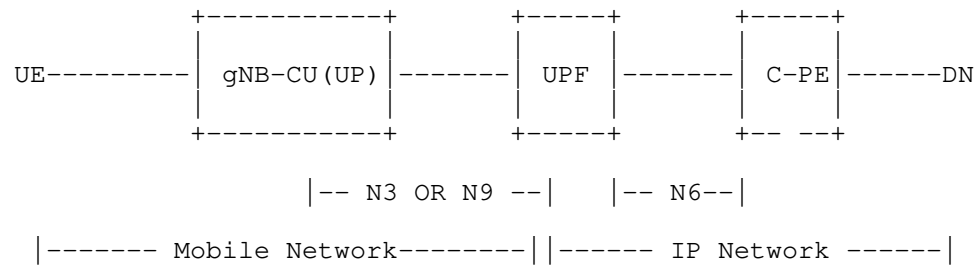
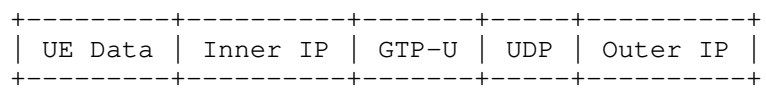


Figure 1: Mobile and IP Data Network for UE

1. UE Packet in the Mobile Network:



2. UE Packet in the IP Network:

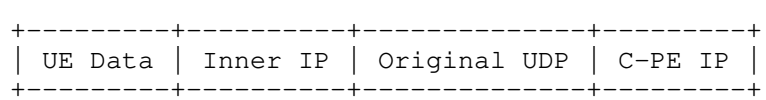


Figure 2: UE Packet Transition from Mobile to IP Network

5. Transport Network Characteristics Mapping to SR-TE Paths

With the 5G Mobile Networking, the UPF would be terminating the mobile connection from the UE. In some Edge Networking scenarios, the UPF would be co-located with the C-PE or it would be connected to the C-PE node over IP Network.

The 5G UE traffic coming to the UPF might be carrying Transport Network Characteristics. In that scenario, there would be a need to maintain Transport Path Characteristic through the core of the network so that end to end SLA can be maintained for the UE traffic.

In scenarios where ingress PE acting as SR-TE node, the mapping of Transport Network Aware Mobility {5G UDP Src Port Range} to {BGP SR-TE Policy, BSID} to be done at the ingress PE. Once this mapping is done, the mobility Transport Path Characteristics can be maintained in the data network.

On a PE node, based on the transport characteristics, the current solution proposes different methods of applying SR-TE path segments:

Scenario 1: In this scenario, the assumption is that the Ingress PE node is connected to the BGP SR-TE Controller through the BGP SR-TE Policy SAFI Session, then this solution defines a mechanism to map the BGP SR-TE Underlay Path Segments based on the Mobility Transport Characteristics.

. This mechanism would require a new BGP Sub-TLV as part of the existing SR Policy SAFI NLRI to download SR-TE Policies corresponds to the mobility Transport Path characteristics. If the TN aware mobility packet UDP Source Port value falls within the UDP Src Port range value of this Sub-TLV, then the pre-downloaded SR-TE Policy MUST be applied on the mobility traffic to map to the correct network slice in the Data Network. Once the Policy is fetched it would be cached by the PE node for operating in-line for the subsequent mobility TN aware packets.

Scenario 2: In this scenario, the assumption is that the Ingress PE node is connected to the SR-PCE (Path Compute Element) Controller through the PCEP Session, then this draft defines a mechanism to map the SR-TE Underlay Segments based on the Mobility Transport Characteristics.

. Currently, this mechanism does not require new encoding in the PCEP based communication, though it needs local Configuration in the PE node to request the SR-TE Paths from the PCEP based Controller based on on-demand TN aware mobility traffic.

Scenario 3: In this scenario, the assumption is that the Ingress PE node is connected to the SR-TE Controller over Restconf/ Netconf or gRPC session. The existing mechanism would be used to download the SR-TE Underlay Path Segments to the PE node based on the Mobility Transport Path Characteristics.

. The Yang Data Model or Protobuf definition is required to define a new Sub-TLV like Scenario 1. The SR-TE Controller would pre-download the SR-TE Policies with the new Sub-TLV in the Ingress PE using the existing session. Once the specific SR-TE Policy is fetched, it would be cached by the Ingress PE to apply for the mobility TN aware traffic in-line to maintain the network characteristics in the Data Network.

5.1. Extend TN Aware Mobility for BGP SR-TE Policy

1) To integrate Transport Network Aware Mobility with BGP SR-TE Policy at the Ingress PE UPF, the Class-map needs to be defined to

classify the incoming mobility traffic with different Transport Path Characteristic.

2) The Ingress PE UPF is assumed to have a BGP SR-TE Policy SAFI connection with the BGP SR-TE Controller. The Mobility traffic destination would resolve in the BGP Peer Next Hop for which SR-TE Policy to be applied to maintain the same network characteristics beyond the mobility domain.

3) A new 5G Metadata Sub-TLV has been defined for existing SR-Policy SAFI with the UDP Source Port Range to identify the SR-TE path based on the Transport Path characteristics.

4) The BGP SR-TE Controller would be programmed with {5G UDP Src Port Range}. That would create internal mapping Table for {5G UDP Src Port Range} < -- > {BGP SR-TE Policy, BSID}.

5) The BGP SR-TE Controller would download the SR-TE Policy in the Ingress PE through the existing BGP SR-Policy SAFI session, and that the BGP update would include an additional 5G Metadata Sub-TLV. The UDP Src Port range in the 5G Metadata Sub-TLV MUST fall within the UDP Source Port range for the SSTs defined by the [TN-AWARE-MOBILITY] draft. If the UDP Src Port range falls outside the range defined by the [TN-AWARE-MOBILITY] draft, then the SR-TE Policy SHOULD be ignored by the Ingress PE.

6) The SR-TE Policy-based traffic steering would be applied in the Ingress PE and it would maintain the local mapping for the reverse Mobility traffic to the UE.

The following class-map definition needs to be applied in the headend PE for the incoming Transport Network aware mobility traffic path:

```
Class-map type traffic match MIOT

     Match UDP Src Port Range Xx - Xy

Class-map type traffic match URLLC

     Match UDP Src Port Range Yx - Yy

Class-map type traffic match EMBB

     Match UDP Src Port Range Zx - Zy
```

The class-map would help to identify the incoming mobility traffic characteristics. Based on these characteristics the headend PE would be able to map the Transport Network aware mobility traffic to the appropriate BGP SR-TE Policy path over the Data Network to reach the UE's destination.

The below figure tries to capture the overall topology, and how to map the mobility traffic in the Ingress PE having BGP SR-Policy SAFI connection with the BGP SR-TE Controller:

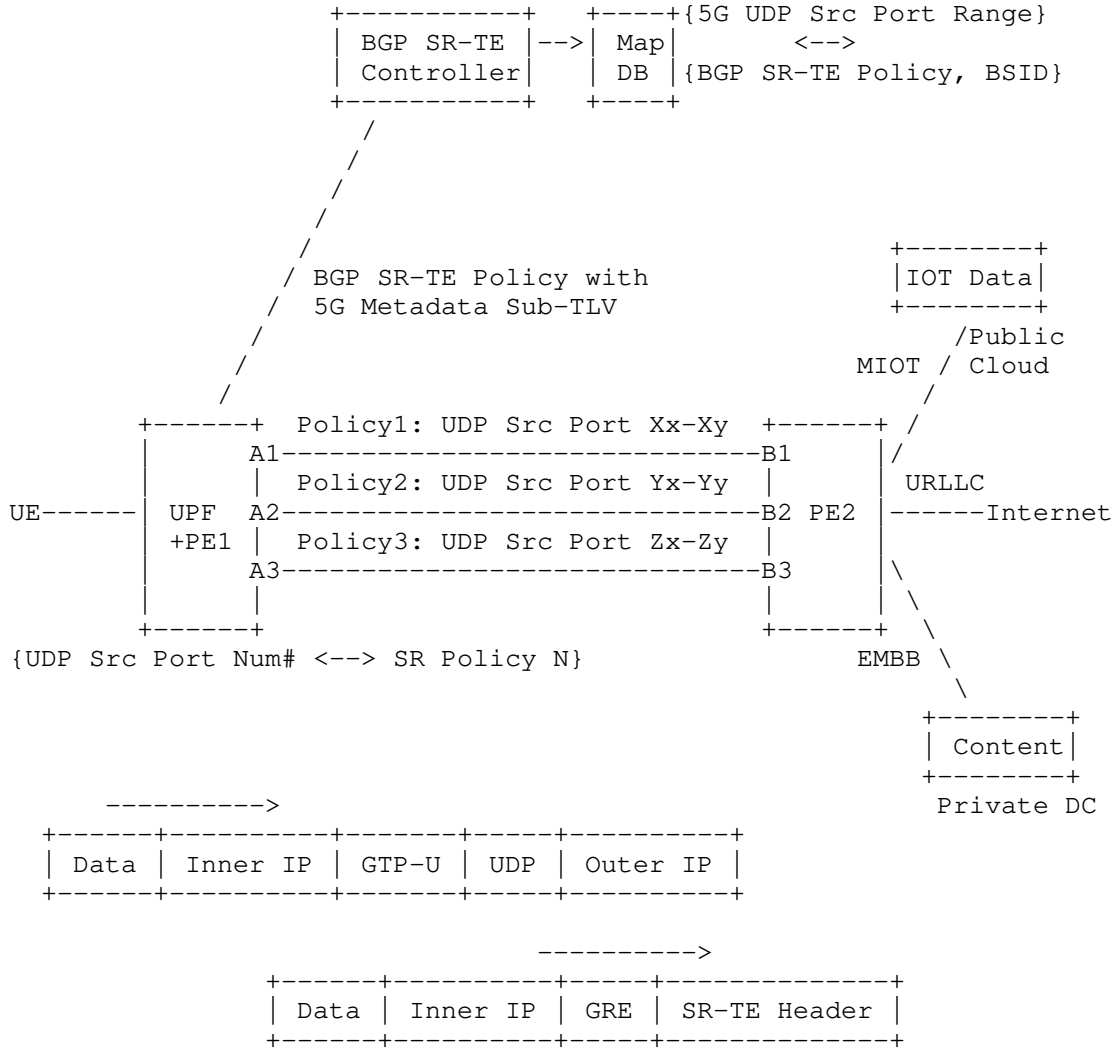


Figure 3: TN Aware Mobility Traffic Mapping to BGP SR-TE Policy Path

Note that, in the above figure the GRE and SR-TE Header is shown as an illustrative purposes and the actual outgoing packet format is based on the SR-TE mechanism (SR-MPLS or SRv6) on the Ingress PE.

To support the Transport Network Mobility Traffic Mapping to BGP SR-TE Policy Path in the headend PE, a new 5G Metadata Sub-TLV needs to be supported. The proposed BGP SR Policy Encoding from the BGP SR-TE Policy Controller to the headend PE node is defined below:

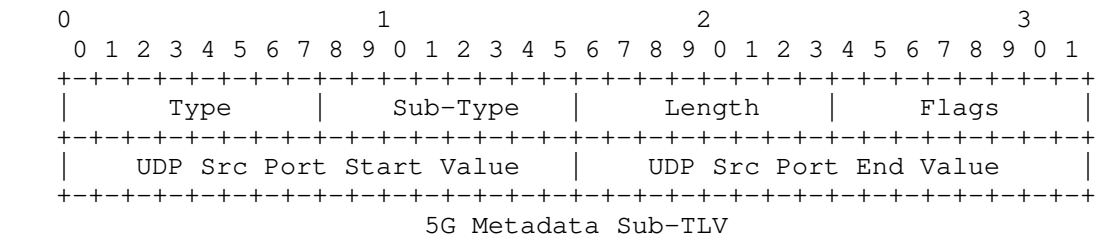
```
SR Policy SAFI NLRI: <Distinguisher, Policy-Color, Endpoint>
   Attributes:
      Tunnel Encap Attr (23)
        Tunnel Type: SR Policy
          Existing Policy Sub-TLV
          5G Metadata Sub-TLV
```

The draft [BGP-SR-TE-POLICY] defines BGP SR-TE Policy encodings. There is no change in the existing encoding that is being used from the BGP SR-TE Controller to the headend PE node. The current solution proposes the new 5G Metadata Sub-TLV for BGP SR-TE Controller to download the SR Policies to the headend PE and to apply the SR-TE Policy-driven path for the Transport Network aware mobility traffic.

The incoming TN aware mobility traffic with UDP Src port and BGP NH to the traffic destination would be used as a key to find the BGP SR-TE Policy. If the BGP Next Hop of the traffic matches with the SR Policy SAFI NLRI Endpoint, and UDP Src Port value falls within the UDP Src Port range defined by the 5G Metadata Sub-TLV, the SR Policy would be applied to the mobility traffic to maintain the traffic characteristics in the data network. The BGP SR-TE Controller would be pre-provisioned with the 5G UDP SRC Port Range based on the [TN-AWARE-MOBILITY] draft, and their corresponding BGP SR-TE Policy.

The 5G Metadata sub-TLV is optional and it MUST NOT appear more than once in the SR-TE Policy.

The format of the new SR-TE 5G Metadata Sub-TLV is captured below:



5G Metadata Sub-TLV

where:

- Type: To be defined by IANA.
- Sub-Type: This field has one of the following values:

  0: Reserved.
  1: UDP Source Port Range.
  2 - 255: Reserved for future use.

- Length: 6 octets.
- Flags: 1 octet of flags. None are defined at this stage. Flags SHOULD be set to zero on transmission and MUST be ignored on receipt.
- UDP Src Port Start Value: 2 octets value to define the staring of the value of the UDP Src Port range.
- UDP Src Port End Value: 2 octets value to define the end value of the UDP Src Port range.

5.2. Extend TN Aware Mobility for SR-PCE Controller

1) To integrate Transport Network Aware Mobility with SR-TE ODN based PCE Controller at the Ingress PE UPF, the Class-map needs to be defined to classify the incoming mobility traffic with different Transport Path Characteristic.

2) The Ingress PE UPF is assumed to have PCEP based communication with the SR-PCE Controller.

3) The Ingress PE would define the Policy-map to map the Transport Path characteristics into SR-TE Color.

4) The Segment Routing TE Configuration for different Metric types will associate the SR-TE Colors with their corresponding TE metric type.

5) The existing SR-TE ODN based PCEP messages with TE metric type and value MUST be used to associate the SR-TE Path corresponding to the 5G UDP Src Port.

6) In this case, the mapping between {5G UDP Src Port} and {SR-TE Policy} would be maintained by the Ingress PE.

7) Once the TN aware mobility traffic destination resolves into a destination of BGP Peer Next Hop, the SR-TE ODN based traffic steering MUST be applied based on the UDP Src Port value of the incoming traffic.

The class-map definition to identify the incoming mobility traffic characteristics is already defined in Section 5.1. The same class-map definition applicable here as well.

The policy-map definition to associate SR-TE color with Transport Path characteristics is defined below:

```
Policy-map type Transport-Network-Aware-Mobility

    Class type traffic MIOT

        Set color <MIOT-10>

    Class type traffic URLLC

        Set color <URLLC-20>

    Class type traffic EMBB

        Set color <EMBB-30>
```

The Segment Routing TE Configuration mechanism can associate the SR-TE Colors with their corresponding metric type. That exists today, and there is no change there. It is captured here to show how TN

aware mobility network characteristics get mapped to different TE metrics through this mechanism.

```
Segment-routing traffic-eng

  On-demand color <MIOT-10> dynamic

     Metric

       Type te

  On-demand color <URLLC-20> dynamic

     Metric

       Type latency

  On-demand color <EMBB-30> dynamic

     Metric

       Type igp
```

As a result, mobility Transport Network aware different traffic characteristics like MIOT, URLLC, or EMBB get to assigned corresponding "te" metric types. To fetch the corresponding SR-TE dynamic path from the SR-PCE Controller based on the "te" metric types exists today.

The below figure tries to capture the overall topology, and how to map the mobility traffic in the headend PE having PCEP connection with the SR-PCE Controller:

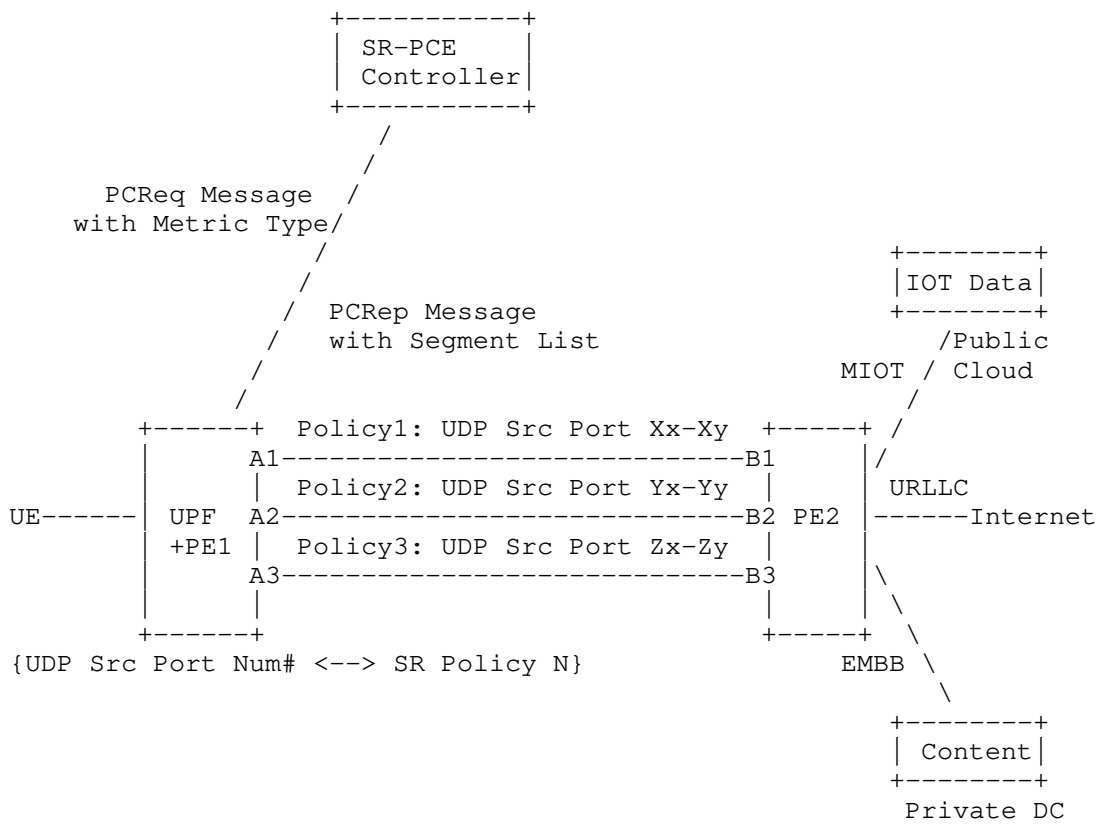


Figure 4: TN Aware Mobility Traffic Mapping to SR-TE Path

5.3. Extend TN Aware Mobility for SR-TE Controller

1) To integrate Transport Network Aware Mobility with SR-TE Policy at the Ingress PE UPF, the Class-map needs to be defined to classify the incoming mobility traffic with different Transport Path Characteristic.

2) The Headend PE UPF is assumed to have Restconf or gRPC connection with the SR-TE Controller. The Mobility traffic destination would resolve in the BGP Peer Next Hop for which SR-TE Policy to be applied to maintain the same network characteristics beyond the mobility domain.

3) A new 5G Metadata Yang data model and Protobuf to be defined for SR-Policy SAFI with UDP Source Port Range to identify the SR-TE path based on the Transport Path characteristics.

4) The SR-TE Controller would be programmed with {5G UDP Src Port Range}. That would create internal mapping Table for {5G UDP Src Port Range} < -- > {BGP SR-TE Policy, BSID}.

5) As the Headend PE sends the 5G metadata Yang data model or Protobuf, the Controller will find a matching SR-TE Policy based on the UDP Source Port.

6) The SR-TE Controller would download the SR-TE Policy in the Ingress PE through the existing Restconf or gRPC session, and that BGP update would include an additional 5G Metadata Sub-TLV. The UDP Src Port range in the 5G Metadata Sub-TLV MUST fall within the UDP Source Port range for the SSTs defined by the [TN-AWARE-MOBILITY] draft. If the UDP Src Port range falls outside the range defined by the [TN-AWARE-MOBILITY] draft, then the SR-TE Policy SHOULD be ignored by the Ingress PE.

7) The SR-TE Policy-based traffic steering would be applied in the Ingress PE UPF and it would maintain the local mapping for the reverse Mobility traffic to the UE.

The class-map definition to identify the incoming mobility traffic characteristics is already defined in Section 5.1. The same class-map definition works here as well.

The below figure tries to capture the overall topology, and how to map the mobility traffic in the headend PE having BGP SR-Policy SAFI connection with the BGP SR-PCE Controller:

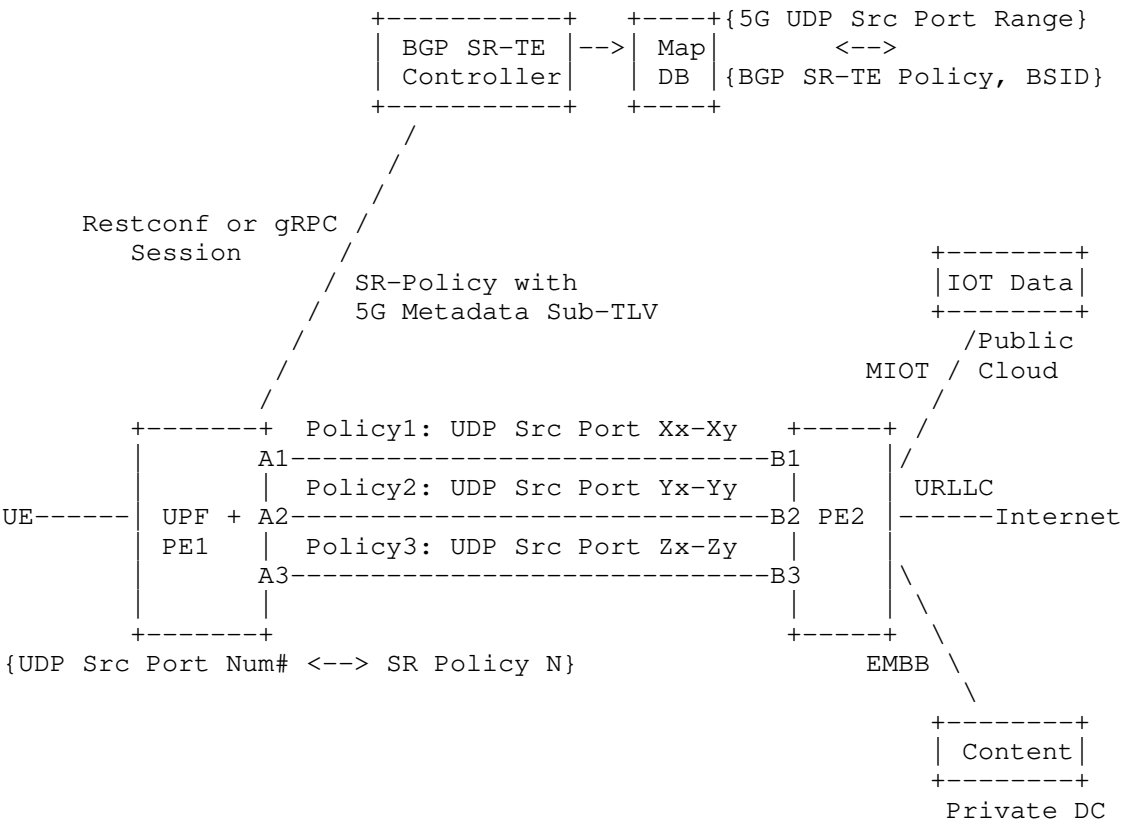


Figure 5: TN Aware Mobility Traffic Mapping to SR-TE Path

6. Mapping of TN Characteristics on SD-WAN Edge Node

On an SD-WAN CE Node, based on the mobility Transport Network characteristics, mapping of mobility aware transport packets to the secure and un-secure tunnel path needs to be achieved.

The [BGP-IPSEC-Discover] draft defines how SD-WAN Edge Node maps the overlay/client routes to the underlay secure tunnel routes.

The current proposal specifies a generic approach on how SD-WAN Edge Node maps the Mobility Transport Network aware traffic to the Secure Tunnels, or Un-Secure TE Paths, or Secure SR-TE Tunnel Paths.

The [SDWAN-BGP-USAGE] draft describes how BGP can be used as a Control Plane for the SD-WAN network and defines the use case for the Hybrid SD-WAN network.

In the case of a hybrid SD-WAN use case, UPF can run part of the SD-WAN edge node or it could be connected to it over an IP network. This would be a use case scenario for Enterprise 5G.

In that scenario, the Transport Path Characteristic for the 5G mobile traffic need to be mapped to Secure (IPSec Tunnel) or Un-secure path (could be MPLS based).

The existing [TN-AWARE-MOBILITY] draft needs to be extended to support new Transport Path Characteristics "Security" for the mobile traffic where security is important for certain mobile traffic.

Based on the UDP Src Port characteristics coming from the mobile network, the SD-WAN edge node would be able to decide what traffic it needs to put in the secure tunnel vs. un-secure tunnel where low latency more important than security.

The below figure tries to capture the overall topology, and how to map the mobility traffic in the SD-WAN Edge Device for Enterprise 5G cases:

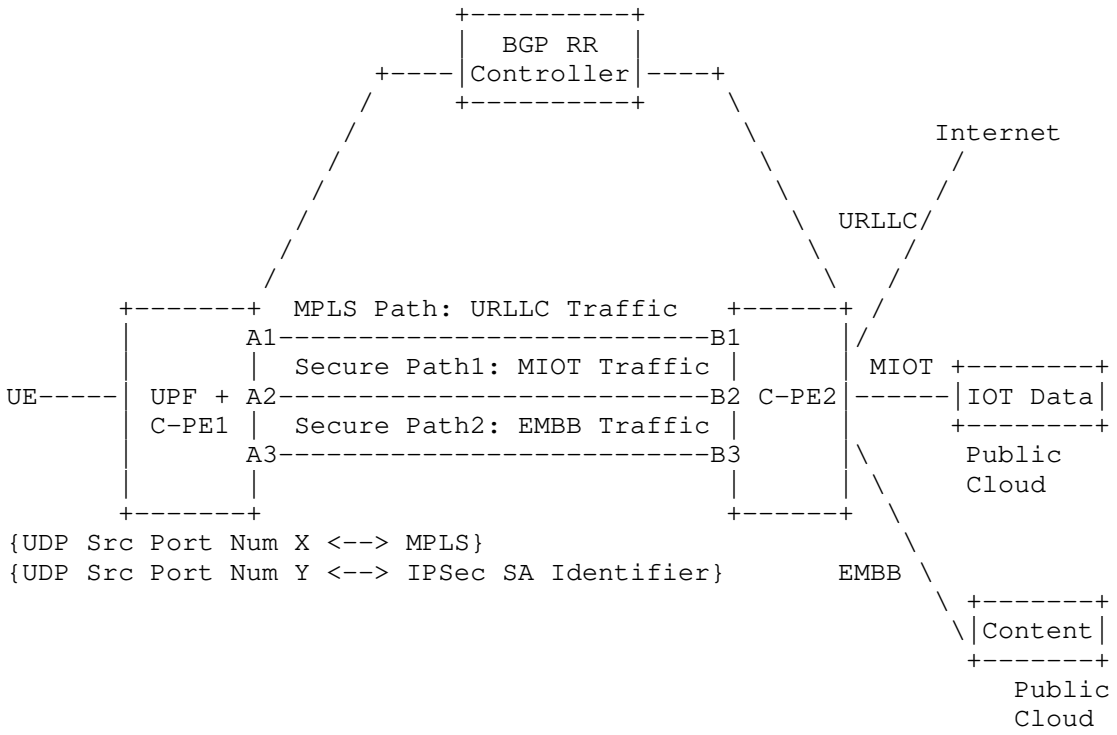


Figure 6: TN Aware Mobility Traffic Mapping in the SD-WAN Edge Device

Here in this diagram, the traffic coming from the mobility side with Transport Network characteristics gets mapped to the underlay un-secure or secure traffic path.

The SD-WAN Edge Node can map the URLLC traffic without any security characteristics to the underlay MPLS path, whereas MIOT, and EMBB traffic with security characteristics gets mapped to the underlay Secure IPSec Tunnel path. The mapping between SD-WAN overlay and underlay routes are described in the [BGP-IPSEC-Discovery] draft.

This solution extends it for Transport Network aware mobility traffic. The SD-WAN Edge Node here identifies the incoming mobility traffic characteristics using the class-map definition, and that is already defined under Section 5.1. Based on the incoming traffic

characteristics, the Edge Node will be able to map the mobility overlay traffic with the SD-WAN underlay tunnel.

7. IANA Considerations

The newly defined 5G Metadata Sub-TLV would need an IANA code point allocation for the Type field. A request for any IANA code point allocation would be submitted.

8. Security Considerations

This document does not introduce any new security issues.

9. References

9.1. Normative References

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

9.2. Informative References

[RFC5440] JP. Vasseur, Ed., JL. Le Roux, Ed., "Path Computation Element (PCE) Communication Protocol (PCEP)", March 2009

[TN-AWARE-MOBILITY] U. Chunduri, et al, "Transport Network aware Mobility for 5G", draft-clt-dmm-tn-aware-mobility-07, April 2021

[BGP-SR-TE-POLICY] S. Previdi, et al, "Advertising Segment Routing Policies in BGP", draft-ietf-idr-segment-routing-te-policy-09, November 2020

[SDWAN-BGP-USAGE] L. Dunber, et al, "BGP Usage for SDWAN Overlay Networks", draft-dunbar-bess-bgp-sdwan-usage-08, January 2021

[BGP-IPSEC-Discover] L. Dunber, et al, "BGP UPDATE for SDWAN Edge Discovery", draft-dunbar-idr-sdwan-edge-discovery-00, January 2021

[Tunnel-Encap] E. Rosen, et al "The BGP Tunnel Encapsulation Attribute", draft-ietf-idr-tunnel-encaps-19, March 2021.

10. Acknowledgments

TBD.

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

Authors' Addresses

Kausik Majumdar
CommScope
350 W Java Drive, Sunnyvale, CA 94089

Email: kausik.majumdar@commscope.com

Uma Chunduri
Futurewei
2330 Central Expressway
Santa Clara, CA 95050

Email: umac.ietf@gmail.com

Linda Dunbar
Futurewei
2330 Central Expressway
Santa Clara, CA 95050

Email: linda.dunbar@futurewei.com

BFD Working Group
Internet-Draft
Updates: 5798 (if approved)
Intended status: Standards Track
Expires: May 5, 2021

G. Mirsky
ZTE Corp.
J. Tantsura
Apstra
G. Mishra
Verizon Inc.

November 1, 2020

Bidirectional Forwarding Detection (BFD) for Multi-point Networks and Virtual Router Redundancy Protocol (VRRP) Use Case draft-mirsky-bfd-p2mp-vrrp-use-case-05

Abstract

This document discusses the use of Bidirectional Forwarding Detection (BFD) for multipoint networks to provide Virtual Router Redundancy Protocol (VRRP) with sub-second Master convergence and defines the extension to bootstrap point-to-multipoint BFD session.

This draft updates RFC 5798.

Status of This Memo

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

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

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

This Internet-Draft will expire on May 5, 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

The [RFC5798] is the current specification of the Virtual Router Redundancy Protocol (VRRP) for IPv4 and IPv6 networks. VRRPv3 allows for a faster switchover to a Backup router. Using such capability with the software-based implementation of VRRP may prove challenging. But it still may be possible to deploy VRRP and provide sub-second detection of Master router failure by Backup routers.

Bidirectional Forwarding Detection (BFD) [RFC5880] had been originally defined detect failure of point-to-point (p2p) paths: single-hop [RFC5881], multihop [RFC5883]. Single-hop BFD may be used to enable Backup routers to detect a failure of the Master router within 100 msec or faster.

[RFC8562] extends [RFC5880] for multipoint and multicast networks, which is precisely characterizes deployment scenarios for VRRP over LAN segment. This document demonstrates how point-to-multipoint (p2mp) BFD can enable faster detection of Master failure and thus minimize service disruption in a VRRP domain. The document also defines the extension to VRRP [RFC5798] to bootstrap a VRRP Backup router to join in p2mp BFD session.

1.1. Conventions used in this document

1.1.1. Terminology

BFD: Bidirectional Forwarding Detection

p2mp: Pont-to-Multipoint

VRRP: Virtual Router Redundancy Protocol

1.1.2. Requirements Language

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. Problem Statement

A router may be part of several Virtual Router Redundancy groups, as Master in some and as Backup in others. Supporting sub-second mode for VRRPv3 [RFC5798] for all these roles without specialized support in data plane may prove challenging. BFD already has many implementations based on HW that are capable of supporting multiple sub-second sessions concurrently.

3. Applicability of p2mp BFD

[RFC8562] may provide an efficient and scalable solution for fast-converging environment that uses the default route rather than dynamic routing. Each redundancy group presents itself as a p2mp BFD session with its Master being the root and Backup routers being tails of the p2mp BFD session. Figure 1 displays the extension of VRRP [RFC5798] to bootstrap tail of the p2mp BFD session. Master

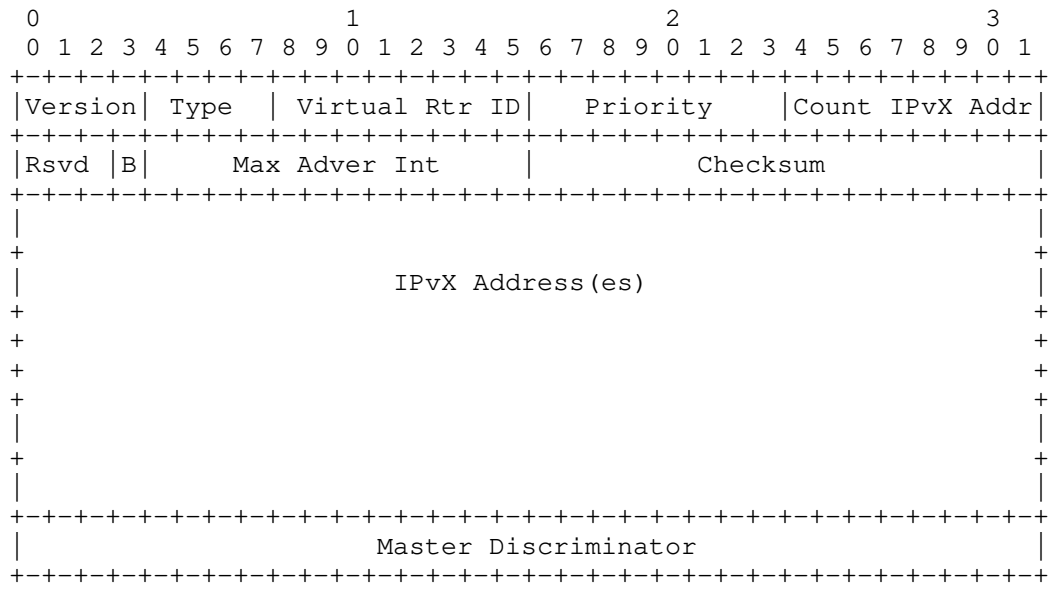


Figure 1: VRRP Extension to Bootstrap P2MP BFD session

where new fields are interpreted as:

B(FD) - a one-bit flag that indicates that the Master Discriminator field is appended to VRRP packet defined in [RFC5798];

Master Discriminator - My Discriminator value allocated by the root of the p2mp BFD session.

The Master router that is configured to use p2mp BFD to support faster convergence of VRRP starts transmitting BFD control packets with VRID as a source IP address and My Discriminator. The same value of My Discriminator MUST be set as the value of the Master Discriminator field. The BFD flag MUST be set in the VRRP packet. Backup router demultiplexes p2mp BFD test sessions based on VRID that it been configured with and the My Discriminator value it learns from the received VRRP packet. When a Backup router detects the failure of the Master router it re-evaluates its role in the VRID. As a result, the Backup router may become the Master router of the given VRID or continue as a Backup router. If the former is the case, then the new Master router MUST select My Discriminator and start transmitting p2mp BFD control packets using Master IP address as the source IP address for p2mp BFD control packets. If the latter is the case, then the Backup router MUST wait for the VRRP packet from the new VRRP Master router that will bootstrap the new p2mp BFD session.

3.1. Multipoint BFD Encapsulation

The MultipointHead of p2mp BFD session when transmitting BFD control packet:

MUST set TTL value to 1 (though note that VRRP packets have TTL set to 255);

SHOULD use group address VRRP ('224.0.0.18' for IPv4 and 'FF02:0:0:0:0:0:0:12' for IPv6) as destination IP address

MAY use network broadcast address for IPv4 or link-local all nodes multicast group for IPv6 as destination IP address;

MUST set destination UDP port value to 3784 when transmitting BFD control packets, as defined in [RFC8562];

MUST use the Master IP address as the source IP address.

4. IANA Considerations

This document makes no requests for IANA allocations. This section may be deleted by RFC Editor.

5. Security Considerations

Security considerations discussed in [RFC5798], [RFC5880], and [RFC8562], apply to this document.

6. Acknowledgements

7. Normative References

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

[RFC5798] Nadas, S., Ed., "Virtual Router Redundancy Protocol (VRRP) Version 3 for IPv4 and IPv6", RFC 5798, DOI 10.17487/RFC5798, March 2010, <https://www.rfc-editor.org/info/rfc5798>.

[RFC5880] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD)", RFC 5880, DOI 10.17487/RFC5880, June 2010, <https://www.rfc-editor.org/info/rfc5880>.

[RFC5881] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for IPv4 and IPv6 (Single Hop)", RFC 5881, DOI 10.17487/RFC5881, June 2010, <https://www.rfc-editor.org/info/rfc5881>.

[RFC5883] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for Multihop Paths", RFC 5883, DOI 10.17487/RFC5883, June 2010, <https://www.rfc-editor.org/info/rfc5883>.

[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>.

[RFC8562] Katz, D., Ward, D., Pallagatti, S., Ed., and G. Mirsky, Ed., "Bidirectional Forwarding Detection (BFD) for Multipoint Networks", RFC 8562, DOI 10.17487/RFC8562, April 2019, <https://www.rfc-editor.org/info/rfc8562>.

Authors' Addresses

Greg Mirsky
ZTE Corp.

Email: gregimirsky@gmail.com

Jeff Tantsura
Apstra

Email: jefftant.ietf@gmail.com

Gyan Mishra
Verizon Inc.

Email: gyan.s.mishra@verizon.com

RTG Working Group
Internet-Draft
Intended status: Standards Track
Expires: May 22, 2021

G. Mirsky
ZTE Corp.
J. Tantsura
Apstra
G. Mishra
Verizon Inc.

November 18, 2020

Bidirectional Forwarding Detection (BFD) on Multi-chassis Link Aggregation Group (MC-LAG) Interfaces
draft-mtm-rtgwg-bfd-mc-lag-01

Abstract

This document describes the use of Bidirectional Forwarding Detection for Multi-chassis Link Aggregation Group to provide faster than Link Aggregation Control Protocol convergence. This specification enhances RFC 7130 "Bidirectional Forwarding Detection (BFD) on Link Aggregation Group (LAG) Interfaces".

Status of This Memo

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

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

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

This Internet-Draft will expire on May 22, 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

The [RFC7130] defines the use of Bidirectional Forwarding Detection (BFD) on Link Aggregation Group (LAG) interfaces. A multi-chassis LAG (MC-LAG) is a type of LAG [IEEE.802.1AX.2008] with member links terminated on separate chassis. [IEEE.802.1AX.2008] does not specify MC-LAG but doesn't preclude it either. Link Aggregation Control Protocol (LACP), also defined in [IEEE.802.1AX.2008], can work with MC-LAG but, as in the LAG case, the fastest link failure detection interval is only in a range of single-digit seconds. This document defines how the mechanism defined to work on LAG interfaces [RFC7130] can be adapted to the MC-LAG case to enable sub-second detection of member link failure.

1.1. Conventions used in this document

1.1.1. Acronyms

BFD: Bidirectional Forwarding Detection

LAG: Link Aggregation Group

LACP: Link Aggregation Control Protocol

MC-LAG: Multi-chassis Link Aggregation Group

MPLS: Multi-Protocol Label Switching

1.1.2. Requirements Language

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. Problem Statement

[RFC7130] does not specify the selection of the destination IP address for the BFD control packet. The only requirement related to the selection is in Section 2.1, stating that the use of the address family across all member links of the given LAG MUST be consistent across all the links. Thus it is implied that the same unicast IP address will be used on all member links of the LAG as the use of different destination addresses would defeat the purpose of [RFC7130] transforming the case into a set of single-hop BFD sessions [RFC5881]. But a single unicast IP address may not work in the MC-LAG case as the member links are terminated on the separate chassis. This document proposes overcoming this problem if using IP or Multi-Protocol Label Switching (MPLS) data plane encapsulation.

3. BFD on MC-LAG with IP-only Data Plane

As described in [RFC7130], a micro-BFD session on the LAG interfaces may use IPv4 or IPv6 address family. In some cases, two sessions, one with IPv4 and one with IPv6 addresses, may run concurrently. This document doesn't change any of these but specifies the selection of the destination IP address in the MC-LAG use case:

- if IPv4 address family is used for the micro-BFD session, then an address from the link-local multicast address 224.0.0.0/24 range SHOULD be used as the destination IP address. The subnet broadcast address MAY be used as the destination IP address as well;

- if the address family used is IPv6, then the IPv6 All Routers address with the link scope, as defined in [RFC4291], FF02::2/128 MUST be used as the destination IP address.

4. BFD on MC-LAG with IP/MPLS Data Plane

IP/UDP is the most natural encapsulation format for the case of micro-BFD on MC-LAG over IP/MPLS data plane as displayed in Figure 1.

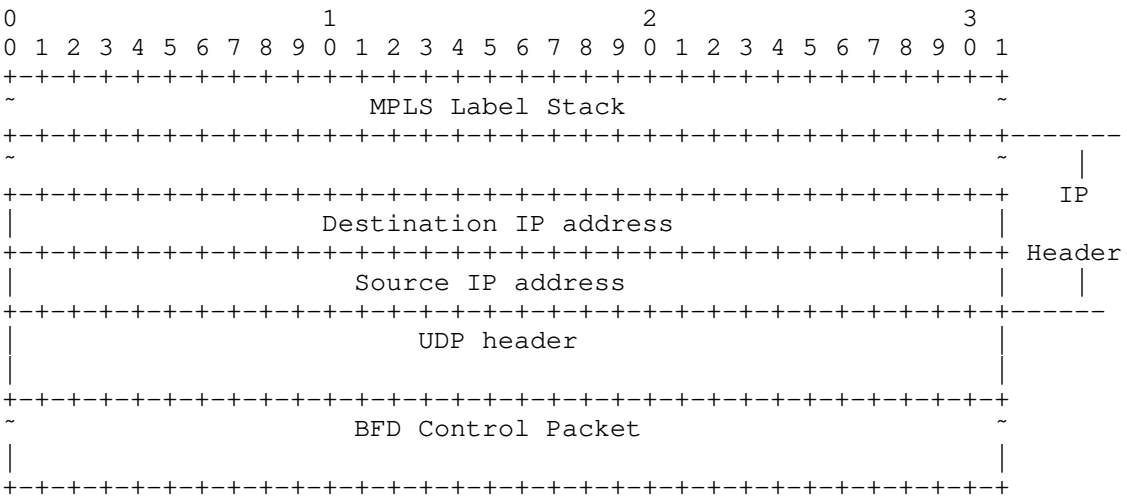


Figure 1: BFD on MC-LAG member link on IPv4/MPLS data plane

An IP and UDP headers immediately follow an MPLS label stack. The destination IP address MUST be set to the loopback address 127.0.0.1/32 for IPv4 [RFC1812], or the loopback address ::1/128 for IPv6 [RFC4291]. TTL or Hop Limit field value MUST be set to 255, according to [RFC5881].

5. IANA Considerations

This document makes no requests for IANA allocations. This section may be deleted by RFC Editor.

6. Security Considerations

This document does not introduce new security concerns but inherits all security considerations discussed in [RFC5881] and [RFC7130].

7. Acknowledgements

TBD

8. References

8.1. Normative References

[IEEE.802.1AX.2008]
"IEEE Standard for Local and metropolitan area networks - Link Aggregation", IEEE 802.1-AX, November 2008.

[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>.

[RFC5881] Katz, D. and D. Ward, "Bidirectional Forwarding Detection (BFD) for IPv4 and IPv6 (Single Hop)", RFC 5881, DOI 10.17487/RFC5881, June 2010, <https://www.rfc-editor.org/info/rfc5881>.

[RFC7130] Bhatia, M., Ed., Chen, M., Ed., Boutros, S., Ed., Binderberger, M., Ed., and J. Haas, Ed., "Bidirectional Forwarding Detection (BFD) on Link Aggregation Group (LAG) Interfaces", RFC 7130, DOI 10.17487/RFC7130, February 2014, <https://www.rfc-editor.org/info/rfc7130>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

8.2. Informative

[RFC1812] Baker, F., Ed., "Requirements for IP Version 4 Routers", RFC 1812, DOI 10.17487/RFC1812, June 1995, <https://www.rfc-editor.org/info/rfc1812>.

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

Authors' Addresses

Greg Mirsky
ZTE Corp.

Email: gregimirsky@gmail.com

Jeff Tantsura
Apstra

Email: jefftant.ietf@gmail.com

Gyan Mishra
Verizon Inc.

Email: gyan.s.mishra@verizon.com

RTGWG
Internet-Draft
Intended status: Standards Track
Expires: May 5, 2021

Q. Xiong
ZTE Corporation
P. Liu
China Mobile

November 1, 2020

The Problem Statement for Precise Transport Networking
draft-xiong-rtgwg-precise-tn-problem-statement-01

Abstract

As described in [I-D.xiong-rtgwg-precise-tn-requirements], the deterministic networks not only need to offer the Service Level Agreements (SLA) guarantees such as low latency and jitter, low packet loss and high reliability, but also need to support the precise services such as flexible resource allocation and service isolation so as to the Precise Transport Networking. However, under the existing IP network architecture with statistical multiplexing characteristics, the existing deterministic technologies are facing long-distance transmission, queue scheduling, dynamic flows and per-flow state maintenance and other controversial issues especially in Wide Area Network (WAN) applications.

This document analyses the problems in existing deterministic technologies to provide precise services in various industries such as 5G networks.

Status of This Memo

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

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

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

This Internet-Draft will expire on May 5, 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

1.1. Overview

5G network is oriented to the internet of everything. In addition to the Enhanced Mobile Broadband (eMBB) and Massive Machine Type Communications(mMTC) services, it also supports the Ultra-reliable Low Latency Communications (uRLLC) services. The uRLLC services cover the industries such as intelligent electrical network, intelligent factory, internet of vehicles, industry automation and other industrial internet scenarios, which is the key demand of digital transformation of vertical domains. These uRLLC services

demand SLA guarantees such as low latency and high reliability and other deterministic and precise properties.

For the intelligent electrical network, there are deterministic requirements for communication delay, jitter and packet loss rate. For example, in the electrical current difference model, a delay of 3~10ms and a jitter variation is no more than 100us are required. The isolation requirement is also important. For example, the automatic operation, control of a process, isochronous data and low priority service need to meet the requirements of hard isolation. In addition to the requirements of delay and jitter, the differential protection (DP) service needs to be isolated from other services.

The industrial internet is the key infrastructure that coordinate various units of work over various system components, e.g. people, machines and things in the industrial environment including big data, cloud computing, Internet of Things (IOT), Augment Reality (AR), industrial robots, Artificial Intelligence (AI) and other basic technologies. For example, automation control is one of the basic application and the the core is closed-loop control system. The control process cycle is as low as millisecond level, so the system communication delay needs to reach millisecond level or even lower to ensure the realization of precise control. There are three levels of real-time requirements for industrial interconnection: factory level is about 1s, and process level is 10~100ms, and the highest real-time requirement is motion control, which requires less than 1ms.

1.2. Motivation

The applications in 5G networks demand much more deterministic and precise properties. But traditional Ethernet, IP and MPLS networks which is based on statistical multiplexing provides best-effort packet service and offers no delivery and SLA guarantee. The deterministic forwarding can only apply to flows with such well-defined characteristics as periodicity and burstiness.

Technologies to provide deterministic service has been proposed to provide bounded latency and jitter based on a best-effort packet network. IEEE 802.1 Time-Sensitive Networking (TSN) has been proposed to provide bounded latency and jitter in L2 LAN networks. According to [RFC8655], Deterministic Networking (DetNet) operates at the IP layer and delivers service which provides extremely low data loss rates and bounded latency within a network domain. However, the existing mechanisms are not sufficient for precise performance such as precise latency, jitter variation, packet loss and more other precise and deterministic properties.

As described in [xiong-rtgwg-precise-networking-requirements], the deterministic networks not only need to offer the Service Level Agreements (SLA) guarantees such as low latency and jitter, low packet loss and high reliability, but also need to support the precise services such as flexible resource allocation and service isolation so as to the Precise Transport Networking. However, under the existing IP network architecture with statistical multiplexing characteristics, the existing deterministic technologies are facing long-distance transmission, traffic scheduling, dynamic flows, per-flow state maintenance and other controversial issues especially in Wide Area Network (WAN) applications.

This document analyses the problems in existing deterministic technologies to provide precise services in various industries such as 5G networks.

2. Conventions used in this document

2.1. Terminology

The terminology is defined as [RFC8655] and [xiong-rtgwg-precise-networking-requirements].

2.2. Requirements Language

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.

3. Problem Statement

3.1. Problem with Traffic Scheduling Mechanisms

As described in [RFC8655], the primary means by which DetNet achieves its QoS assurances in IP networks is to eliminate the latency and packet loss by the provision of sufficient resource at each node. But only the resource itself is not sufficient, the traffic scheduling mechanisms such as queuing, shaping, and scheduling functions must be applied in L3 networks.

The congestion control, queue scheduling and other traffic mechanisms which have been proposed in IEEE 802.1 TSN. But most of them are based on the time synchronization and time cycle, such as IEEE802.1Qbv, IEEE802.1Qch and so on. It will be difficult to achieve precise time synchronization with all network nodes due to deployment and cost reasons. And the shaping and queuing methods

which are not based on time synchronization such as IEEE802.1Qav and IEEE802.1Qcr might not be suitable or available for some L3 networks such as WAN application where multiple dynamic traffic flows may be existed.

Moreover, the requirements of the all nodes in WAN networks to apply the time synchronization and traffic scheduling mechanisms will also lead to the difficulty of network scalability and deployment.

3.2. Problem with Long-distance Transmission Delay and Jitter

In WAN application, long-distance transmission will lead to uncertainties, such as increasing transmission delay, jitter and loss. The link delay of transmission is variable and can not be ignored, and it must be considered in the end-to-end deterministic forwarding mechanisms which are based on time synchronous. So the following problems should be considered.

Precise measurement of the link delay.

The symmetry of bidirectional forwarding of the link delay.

Time cycle alignment in flows aggregation scenario.

3.3. Problem with SLA Guarantees of Dynamic Flows

As described in [RFC8557], deterministic forwarding can only apply to flows with such well-defined characteristics as periodicity and burstiness. As defined in DetNet architecture [RFC8655], the traffic characteristics of an App-flow can be CBR (constant bit rate) or VBR (variable bit rate) of L1, L2 and L3 layers (VBR takes the maximum value when reserving resources). But the current scenarios and technical solutions only consider CBR flow, without considering the coexistence of VBR and CBR, the burst and aperiodicity of flows. The operations such as shaping or scheduling have not been specified. Even TSN mechanisms are based on a constant and forecastable traffic characteristics.

It will be more complicated in WAN applications where much more flows coexist and the traffic characteristics is more dynamic. It is required to offer reliable delivery and SLA guarantee for dynamic flows. For example, periodic flow and aperiodic flow (including micro burst flow, etc.), CBR and VBR flow, flow with different periods or phases, etc. When the network needs to forward these deterministic flows at the same time, it must solve the problems of time window selection, queue processing and aggregation of multiple flows.

Moreover, the existing solutions do not consider the characteristics analysis of service requirements, including the impact of dynamic characteristics analysis on the network, mainly about how to ensure the certainty in the case of dynamic flows.

3.4. Problem with Service Isolation

In some scenarios, such as intelligent electrical network, the isolation requirements are very important. For example, the automatic operation or control of a process or isochronous data and low priority service need to meet the requirements of hard isolation. In addition to the requirements of delay and jitter, the differential protection (DP) service needs to be isolated from other services and hard isolated tunnel is required.

The resource reservation in DetNet can only ensure the statistical reuse of bandwidth resources, but it can not guarantee the precise isolation and control of instantaneous burst and can not realize the hard isolation of each flow. The existing solutions cannot achieve the requirements of service isolation.

3.5. Problem with Precise Resource Allocation

As described in [RFC8655], the primary means by which DetNet achieves its QoS assurances is to reduce, or even completely eliminate, packet loss by the provision of sufficient buffer storage at each node. But it can not be achieved by not enough resource which can be allocated due to practical and cost reason. The existing solutions can not achieve the precise resource allocation.

4. Security Considerations

TBA

5. Acknowledgements

TBA

6. IANA Considerations

TBA

7. Normative References

[I-D.xiong-rtgwg-precise-tn-requirements]
Xiong, Q. and P. Liu, "The Requirements for Precise Transport Networking", draft-xiong-rtgwg-precise-tn-requirements-00 (work in progress), April 2020.

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8557] Finn, N. and P. Thubert, "Deterministic Networking Problem Statement", RFC 8557, DOI 10.17487/RFC8557, May 2019, <https://www.rfc-editor.org/info/rfc8557>.

[RFC8655] Finn, N., Thubert, P., Varga, B., and J. Farkas, "Deterministic Networking Architecture", RFC 8655, DOI 10.17487/RFC8655, October 2019, <https://www.rfc-editor.org/info/rfc8655>.

Authors' Addresses

Quan Xiong
ZTE Corporation
No.6 Huashi Park Rd
Wuhan, Hubei 430223
China

Email: xiong.quan@zte.com.cn

Peng Liu
China Mobile
Beijing 100053
China

Email: liupengyjy@chinamobile.com

RTGWG
Internet-Draft
Intended status: Standards Track
Expires: May 5, 2021

Q. Xiong
ZTE Corporation
P. Liu
China Mobile

November 1, 2020

The Requirements for Precise Transport Networking
draft-xiong-rtgwg-precise-tn-requirements-01

Abstract

The future networks not only need to offer the Service Level Agreements (SLA) guarantees such as low lantency and jitter, low packet loss and high reliability, but also need to support the precise services such as flexible resource allocation and service isolation. This document proposes a set of performance requirements and precise properties for Precise Transport Networking in various industries such as 5G networks.

Status of This Memo

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

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

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

This Internet-Draft will expire on May 5, 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

5G network is oriented to the internet of everything. In addition to the Enhanced Mobile Broadband (eMBB) and Massive Machine Type Communications(mMTC) services, it also supports the Ultra-reliable Low Latency Communications (uRLLC) services. The uRLLC services cover the industries such as intelligent electrical network, intelligent factory, internet of vehicles, industry automation and other industrial internet scenarios, which is the key demand of digital transformation of vertical domains. These uRLLC services demand SLA guarantees such as low latency and high reliability and other deterministic and precise properties.

For the intelligent electrical network, there are deterministic requirements for communication delay, jitter and packet loss rate. For example, in the electrical current difference model, a delay of 3~10ms and a jitter variation is no more than 100us are required. The isolation requirement is also important. For example, the automatic operation, control of a process, isochronous data and low priority service need to meet the requirements of hard isolation. In addition to the requirements of delay and jitter, the differential protection (DP) service needs to be isolated from other services.

The industrial internet is the key infrastructure that coordinate various units of work over various system components, e.g. people,

machines and things in the industrial environment including big data, cloud computing, Internet of Things (IOT), Augment Reality (AR), industrial robots, Artificial Intelligence (AI) and other basic technologies. For example, automation control is one of the basic application and the the core is closed-loop control system. The control process cycle is as low as millisecond level, so the system communication delay needs to reach millisecond level or even lower to ensure the realization of precise control. There are three levels of real-time requirements for industrial interconnection: factory level is about 1s, and process level is 10~100ms, and the highest real-time requirement is motion control, which requires less than 1ms.

The future networks not only need to offer the Service Level Agreements (SLA) guarantees such as low latency and jitter, low packet loss and high reliability, but also need to support the precise services such as flexible resource allocation and service isolation. This document proposes a set of performance requirements and precise properties for Precise Transport Networking in various industries such as 5G networks.

2. Conventions used in this document

2.1. Terminology

The terminology is defined as [RFC8655].

2.2. Requirements Language

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.

3. Terms of Precise Transport Networking

IEEE 802.1 Time-Sensitive Networking (TSN) has been proposed to provide bounded latency and jitter in L2 LAN networks. According to [RFC8655], Deterministic Networking (DetNet) operates at the IP layer and delivers service with extremely low data loss rates and bounded latency.

However, under the existing IP network architecture with statistical multiplexing characteristics, the existing deterministic technologies are facing long-distance transmission, queue scheduling, dynamic flows and other controversial issues as described in [xiong-rtgwg-precise-tn-problem-statement]. And besides precise latency, jitter, and packet loss, more other precise and deterministic properties and

performances should be provided such as flexible resource allocation and service isolation and so on.

Precise Transport Networking is defined to provide precise SLA guarantees such as latency, jitter, packet loss rate, reliability, and precise control such as flexible resource allocation and service isolation and more other precise services intelligently and dynamically. The purpose of the Precise Transport Networking is based on the hierarchical structure of the transport network, taking advantage of the existing technologies including the flexible precise tunnels technology and the deterministic mechanisms, to support the end-to-end precise service through the characteristics of slicing pieces, hard isolation and preemption characteristics, so as to achieve the high-precision of the future networks.

4. Requirements of Precise Transport Networking

4.1. Precise Latency, Jitter, and Packet Loss

It is required to provide precise Latency, jitter and packet loss dynamically and flexibly in all scenarios for each characterizd flow.

The precise requirements of latency includes bounded latency and low latency. The precise requirements of jitter includes bounded jitter and low jitter. So the precise requirements of latency and jitter may be the combination of latency and jitter, typically including bounded latency and low jitter, low latency and bounded latency, and so on.

4.2. Precise SLA Guarantees for Converged Networks

It is required to provide precise SLA guarantees for converged networks including computing and network convergence, lossless and network convergence, etc.

In some scenarios, such as MEC, it is required to provide precise computing for Controlized CFN/APN. Other resources such as computing resources, energy consumption should be considered. And the utilization and optimization of network resources are extremely important.

4.3. Precise Resource Allocation

As described in [RFC8655], the primary means by which DetNet achieves its QoS assurances is to reduce, or even completely eliminate, packet loss by the provision of sufficient buffer storage at each node. But it can not be achieved by not sufficient resource which can be

allocated due to practical and cost reason. The existing solutions can not achieve the precise resource allocation.

Precise resource allocation is required along with the explicit path with more SLA guarantee parameters like bandwidth, latency, packet loss and so on. The existing technologies such as FlexE and SR tunnels should be taken into consideration.

4.4. Precise Service Isolation

It is required to provide precise service isolation for every flow. In some scenarios, such as intelligent electrical network, the isolation requirements are very important. For example, the automatic operation or control of a process or isochronous data and service with different priorities need to meet the requirements of hard isolation. In addition to the requirements of delay and jitter, the differential protection (DP) service needs to be isolated from other services and hard isolated tunnel is required.

4.5. Precise OAM

It is required to consider precise service performance detection and perception, service support and recovery mechanisms, such as millisecond level service monitoring, 0.0001% packet loss awareness, etc. The existing solutions also do not consider the statistics, analysis and reporting of service performance.

Precise OAM is required including service monitoring, perception, performance statistics, precise service support and recovery mechanism, etc. The OAM mechanisms should be taken into consideration such as In-band OAM, iOAM and so on.

5. Security Considerations

TBA

6. Acknowledgements

TBA

7. IANA Considerations

TBA

8. Normative References

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8655] Finn, N., Thubert, P., Varga, B., and J. Farkas, "Deterministic Networking Architecture", RFC 8655, DOI 10.17487/RFC8655, October 2019, <https://www.rfc-editor.org/info/rfc8655>.

Authors' Addresses

Quan Xiong
ZTE Corporation
No.6 Huashi Park Rd
Wuhan, Hubei 430223
China

Email: xiong.quan@zte.com.cn

Peng Liu
China Mobile
Beijing 100053
China

Email: liupengyjy@chinamobile.com