rtgwg
Internet-Draft
Intended status: Informational
Expires: 9 January 2023

Y. Li
L. Iannone
D. Trossen
Huawei Technologies
P. Liu
China Mobile
C. Li
Huawei Technologies

8 July 2022

Dynamic-Anycast Architecture
draft-li-dyncast-architecture-04

Abstract

This document describes a proposal for an architecture for the Dynamic-Anycast (Dyncast). It includes an architecture overview, main components that shall exist, and the workflow. An example of workflow is provided, focusing on the load-balance multi-edge based service use-case, where load is distributed in terms of both computing and networking resources through the dynamic anycast architecture.

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 9 January 2023.

Table of Contents

1. Introduction 2
2. Definition of Terms 3
3. Architecture Main Concepts 4
4. Dyncast Architecture Workflow 8
4.1. Service Notification/Metrics Update 8
4.2. Service Demand Dispatch and Instance Affinity 10
5. Dyncast Control-plane vs Data-plane operations 11
6. Summary 12
7. Security Considerations 12
8. IANA Considerations 12
9. Contributors 12
10. Informative References 13
Acknowledgements 13
Authors' Addresses 13

1. Introduction

Edge computing has been expanding from single edge nodes to multiple networked collaborating edge nodes to solve the issues like response time, resource optimization, and network efficiency.

The current network architecture in edge computing provides relatively static service dispatching, often to the closest edge from an IGP perspective, or to the server with the most computing resources without considering the network status, and even sometimes just based on static configuration.

Traffic steering that takes into account computing resource metrics seems to be an interesting paradigm that would benefit several use-cases [I-D.liu-dyncast-ps-usecases]. Yet, more investigation is still needed in key areas for this paradigm and, to this end, this document aims at providing an architectural framework, which will

enable compute- and network-aware traffic steering decisions in edge computing.

The Dyncast architecture presents an anycast based service and access model addressing 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.

Dyncast assumes that there are multiple equivalent service instances running on different edge nodes, globally providing (from a logical point of view) one single service. A single edge may have limited computing resources available, and different edges likely have different resources available, such as CPU or GPU. The main principle of Dyncast is that multiple edge nodes are interconnected and collaborate with each other to achieve a holistic objective, namely to dispatch service demands taking into account both service instances status as well as network state (e.g., paths length and their congestion). For this, computing resources available to serve a request is one of the top metrics to be considered. At the same time, the quality of the network path to an edge node may vary over time and may hence be another key attribute to be considered for said dispatching of service demands.

2. Definition of Terms

Dyncast: As defined in [I-D.liu-dyncast-ps-usecases], Dynamic Anycast, taking the dynamic nature of computing resource metrics into account to steer an anycast routing decision.

Service: As defined in [I-D.liu-dyncast-ps-usecases], a monolithic functionality that is provided by an endpoint according to the specification for said service. A composite service can be built by orchestrating monolithic services.

Service instance: As defined in [I-D.liu-dyncast-ps-usecases], running environment (e.g., a node) that makes the functionality of a service available. One service can have several instances running at different network locations.

D-Router: A node supporting Dyncast functionalities as described in this document. Namely it is able to understand both network-related and service-instances-related metrics, take forwarding decision based upon and manitain instance affinity, i.e., forwards packets belonging to the same service demand to the same instance.

D-MA: Dyncast Metric Agent (D-MA): A dyncast specific agent able to gather and send metric updates (from both network and instance prespective) but not performing forwarding decisions. May run on a D-Router, but it can be also implementated as a separate module (e.g., a software library) collocated with a service instance.

D-SID: Dyncast Service ID, an identifier representing a service, which the clients use to access said service. Such identifier identifies all of the instances of the same service, no matter on where they are actually running. D-SID is independent of which service instance serves the service demand. Usually multiple instances provide a (logically) single service, and service demands are dispatched to the different instance through an anycast model, i.e., choosing one instance among all available instances.

D-BID: Dyncast Binding ID, an address to reach a service instance for a given D-SID. It is usually a unicast IP where service instances are attached. Different service instances provide the same service identified through D-SID but with different Dyncast Binding IDs.

Service demand: The demand for a specific service and addressed to a specific D-SID.

Service request: The request for a specific service and addressed to a specific service instance identified with D-BID.

3. Architecture Main Concepts

Dyncast assumes that there are multiple equivalent service instances running on different edge sites, globally providing one single service which is represented by D-SID. The network will take forwarding decision for the service demand from the client according to both service instances status as well as network state.

The architecture of Dyncast has two typical modes, distributed or centralized.

* Distributed mode: The resources and status of the different service instances are propagated from the D-Routers connecting the edge sites where the service is deployed to the D-Routers with clients. In addition D-Routers have the network topology and status. The ingress D-Router which receives the service demand from the client decides independently which service instance to access according to the service instances status and network state and maintains instance affinity.

* Centralized mode: The resources and status of the different service instances are reported to the network controller from the D-Routers connecting the edge sites where the service is deployed. At the same time the controller collects the network topology and status. The controller makes routing decisions for each ingress D-Router according to the service instances status and network state and downloads the decisions to all the ingress D-Routers.

When the ingress D-Router receives the service demand from the client, it selects which service instance to access according to the decision made by the controller, and maintains the instance affinity subsequently.

This document mainly introduces the detailed process of the distributed mode, and the centralized mode will be introduced in detail in the future.

Edge sites (edges for short) are normally the sites where edge computing is performed. Service instances are initiated at different edge sites. Thus, a single service can actually have a significant number of instances running on different edges. A Dyncast Service ID (D-SID) is used to uniquely identify a service (e.g., a matrix computation for face recognition, or a game server). Service instances can be hosted on servers, virtual machines, access routers or gateway in edge data center.

Close to (one or more) Service instances is the Dyncast Metric Agent (D-MA). This element has the task to gather information about resources and status of the different instances as well as network-related information. Such element may also run in a dyncast-enable

router (named D-Router), while other deployment scenarios may lead to this element running separately on edge nodes.

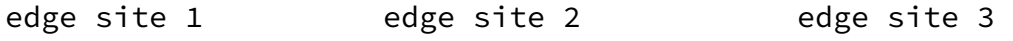
A D-Router is actually the main element in a Dyncast network, providing the capability to exchange the information about the computing resources information of service instances which have been gathered through D-MAs. A D-Router can also be a service access point for clients. When a service demand arrives, it will be delivered to the most appropriate service instance. A service demand may be the first packet of a data flow rather than an explicit out of band service request. This architectural document does not make any specific assumption on this matter. This documents only assumes that:

* D-Routers are able to identify new service demands. The Dyncast architecture presented in this document allows then to deliver such a packet to the most appropriate service instance according to information received from D-MAs and other D-Routers.

* D-Router are able to identify packets belonging to an existing service demand. The Dyncast architecture presented in this document allows to deliver these packets always to the same service instance selected at the initial service demand. We term this capability as 'instance affinity'.

Note: As described above, D-Router can make decision based on per-service-instance computing-aware information. Actually, the D-Router can make the decison based on per-site computing-aware information. In this case, the egress D-Router can send the packet to the specific instance based on local policy, Load balancing, etc. This will be described in the future.

The element introduced above are depicted in Figure 1, which shows the proposed Dyncast architecture. In Figure 1, the "infrastructure" indicates the general IP infrastructure that does not necessarily need to support Dyncats, i.e., not all routers of the infrastructure need to be D-Routers.



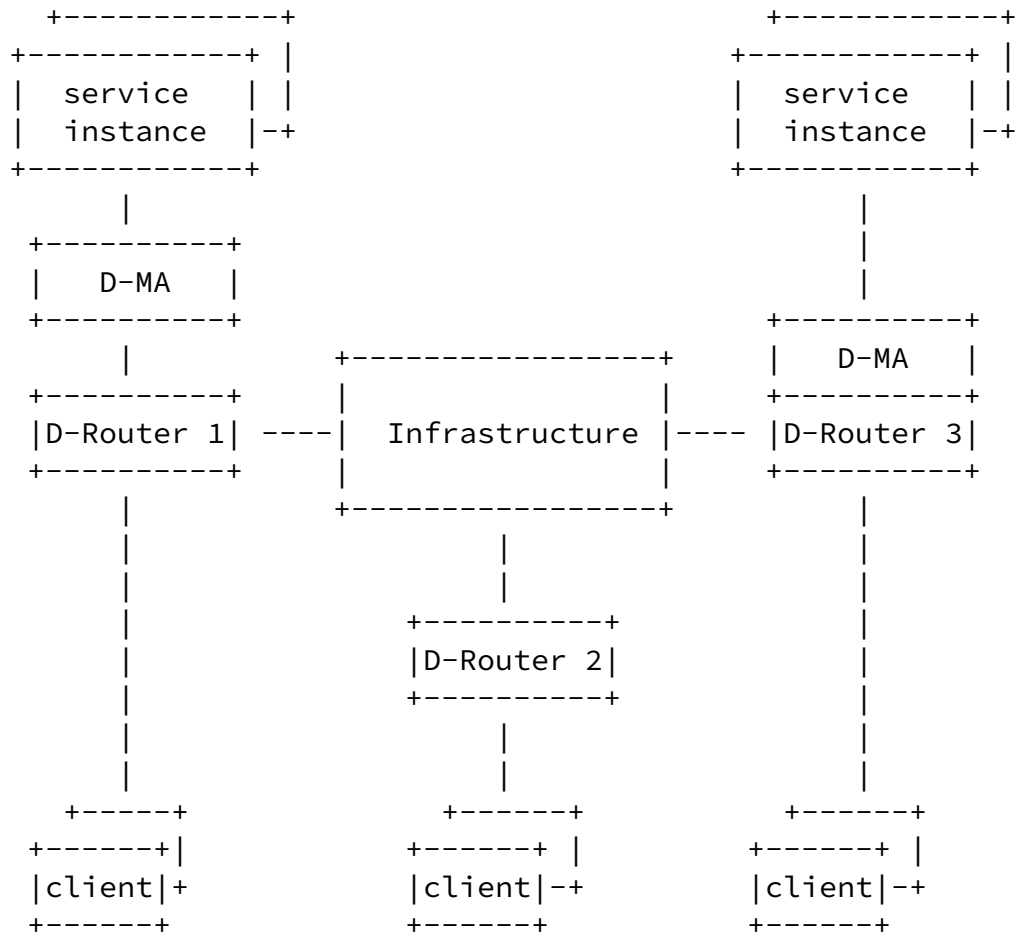


Figure 1: Dyncast Architecture.

Figure 2 shows an example of Dyncast deployment, with 2 service instantiated twice (2 instances) on two different edges, namely edge site 2 and 3. Those service instances utilize different D-BIDs to serve service demands. D-Router 1 doesn't connect the edge site

directly and needn't collect the metric updates by D-MA. But it has client to access and need to take forwarding decision for the client. D-Router 2 gets metric updates by D-MA which runs on it. Edge site 2 has client present, so D-Router 2 need to take forwarding decision. D-Router 3 gets metric updates from D-MA which is a separate software module on edge computing platform in edge site 3. No client is present at edge site 3, so D-Router 3 doesn't need take forwarding decision.

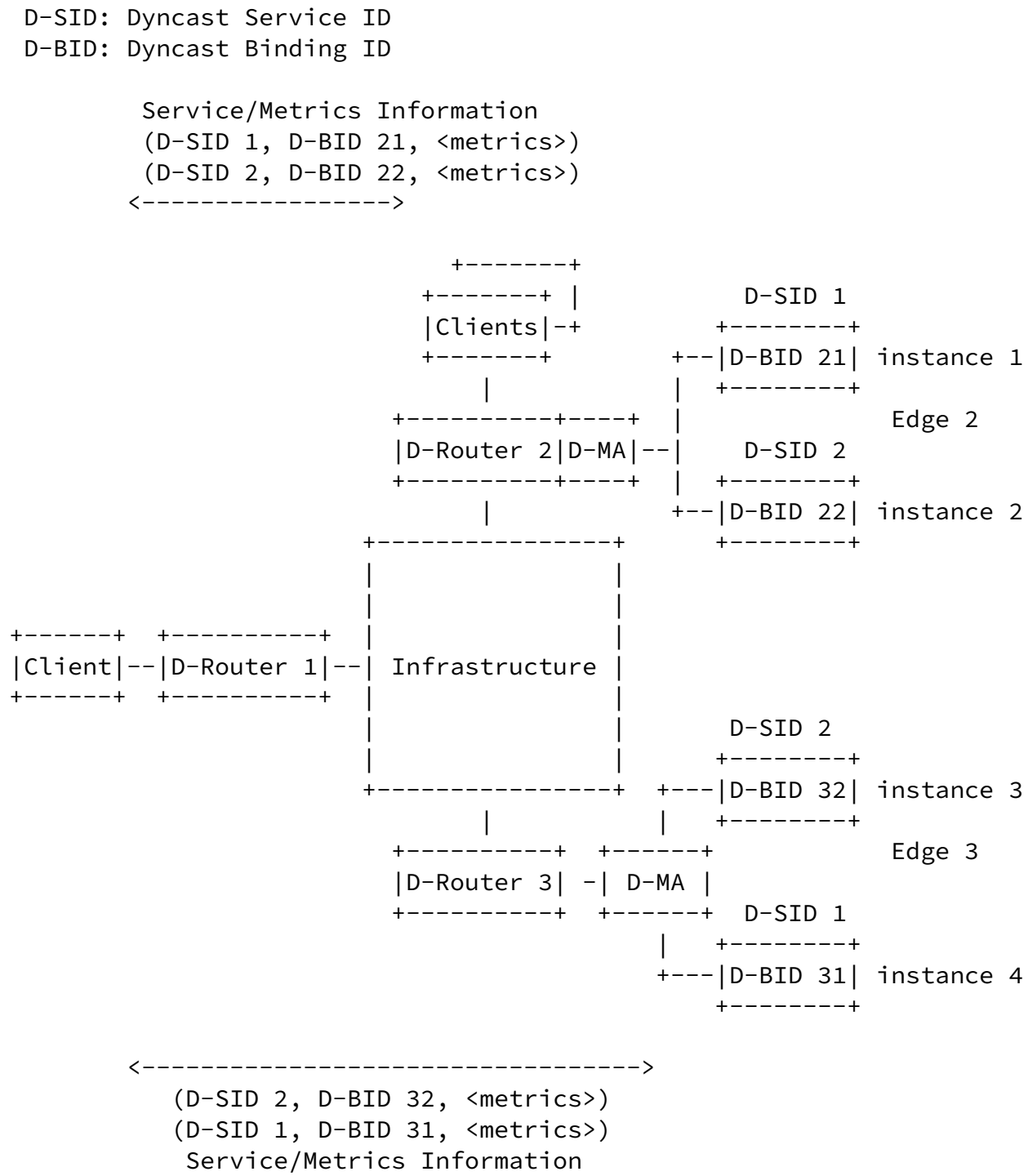


Figure 2: Dyncast deployment example.

In Figure 2, the Dyncast Service ID (D-SID) follows an anycast semantic, such as provided through an IP anycast address. It is used to access a specific service no matter which service instance eventually handles the service demand of the client. Clients or other entities which want to access a service need to know about its D-SID in advance. It can be achieved in different ways, for example, using a special range of addresses associated to a certain service or coding of anycast IP address as D-SID, or using DNS.

The Dyncast Binding ID (D-BID) is a unicast IP address. It is usually the interface IP address through to reach a specific service instance. Mapping and binding a D-SID to a D-BID is dynamic and depends on the computing and network status at the time the service demand first arrives (see Section 4.1 for the reporting of such status). To ensure instance affinity, D-Routers are requested to remember the instance that has been selected (e.g., by storing the mapping) for delivering all packets to the same instance (see Section 4.2 for discussing this aspect).

4. Dyncast Architecture Workflow

The following subsections provide an overview of how the architectural elements introduced in the previous section do work together.

4.1. Service Notification/Metrics Update

When a service instance is instantiated/terminated the service information consisting in the mapping between the D-SID and the D-BID has to be updated/deleteted as well. An update can also be triggered by a change in relevant metrics (e.g., an instance becomes overloaded). Computing resource information of service instance is key information in Dyncast. Some of them may be relatively static like CPU/GPU capacity, and some may be very dynamic, for example, CPU/GPU utilization, number of sessions associated, number of queuing requests. Changes in service-related relevant information has to be collected by D-MA associated to each service instance. Various ways can be used, for example, via routing protocols like EBGP or via an API of a management system. Conceptually a D-Router collects information coming from D-MA and keeps track of the IDs and computing metrics of all service instances. The rate for metrics update depends on the specific algorithm and is out of scope of this document. The update will be sync up only among related D-routers, and will not affect other routers/devices in the network.

Figure 2 shows an example of information shared by the Dyncast elements. The D-MA which is deployed with D-Router2 shares binding information concerning the two instances of the two services running on edge 2 (upper right hand side of the figure). These information is:

* (D-SID 1, D-BID 21, metrics)

* (D-SID 2, D-BID 22, metrics)

The D-MA which is deployed as a separate module on edge 3 (lower right hand side of the figure) shares binding information concerning the two instances of the two services running on edge 3. These information is:

* (D-SID 1, D-BID 31, metrics)

* (D-SID 2, D-BID 32, metrics)

Dyncast nodes share among themselves the service information including the associated computing metrics for the service instances attached to them. As a network node, a D-Router can also monitor the network cost or metrics (e.g., congestion) to reach other D-Routers. This is the focus of Dyncast control plane. Different mechanisms can be used to share such information, for instance BGP ([RFC4760]), an IGP, or a controller based mechanism. The specific mechanism is beyond the scope of this document. The architecture assumes that the Dyncast elements are able to share relevant information.

If, for instance, the client on the left hand side of Figure 2 sends a service demand for D-SID1, D-Router1 has the knowledge of the status of the service instance on both edge 2 and edge 3 and can make a decision toward which D-BID to forward the demand.

There are 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 used for simplicity reasons. However, it may not accurately reflect the computing resources of interest. Multi-dimensional values give finer information. This architectural document does not make any specific assumption about metrics and how to encode or even use them. As stated in Section 3, the only assumption is that a D-Router is able to use such metrics so to take a decision when a service demand arrives in order to map the demand onto a suitable service request.

As explained in the problem statement document

[I-D.liu-dyncast-ps-usecases], computing metrics may change very frequently, when and how frequent such information should be

exchanged among Dyncats elements should be determined also in accordance with the distribution protocol used for such purpose. A spectrum of approaches can be employed,such as interval based updates, threshold triggered updates, policy based updates, etc.

4.2. Service Demand Dispatch and Instance Affinity

This is the focus of the Dyncast data plane. When a new flow (representing a service demand) arrives at a Dyncast ingress, such ingress node selects the most appropriate egress according to the network status and the computing resource of the attached service instances.

Instance affinity is one of the key features that Dyncast should support. It means that packets from the same 'flow' for a service should always be sent to the same egress to be processed by the same service instance. The affinity is determined at the time of newly formulated service demand.

It is worth noting that different services may have different notions of what constitutes a 'flow' and may thus identify a flow differently. Typically a flow is identified by the 5-tuple value. However, 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 by the same service instance. Therefore a 3-tuple based flow identifier is more suitable for this case. Hence, it is desired to provide certain level of flexibility in identifying flows, or from a more general perspective, in identifying the set of packets for which to apply instance affinity. More importantly, the means for identifying a flow for the purpose of ensuring instance affinity must be application-independent to avoid the need for service-specific instance affinity methods.

Specifically, Instance affinity information should be configurable on a per-service basis. For each service, the information can include the flow/packets identification type and means, affinity timeout value, and etc. For instance, the affinity configuration can indicate what are the values, e.g., 5-tuple or 3-tuple, to be used as

the flow identifier.

When the most appropriate egress and service instance is determined when a new flow for a service demand arrives, a binding table should save this association between new service demand and service instance selection. The information in such binding table may include flow/packets identification, affinity timeout value, etc. The subsequent packets matching the entry are forwarded based on the table. Figure 3 shows a possible example of flow binding table at the ingress D-Router.

Flow/Packets Identifier					D-BID egress	timeout
src_IP	dst_IP	src_port	dst_port	proto		
X	D-SID 2	-	8888	tcp	D-BID 32	xxx
Y	D-SID 2	-	8888	tcp	D-BID 12	xxx

Figure 3: Example of what a binding table can look like.

5. Dyncast Control-plane vs Data-plane operations

In summary, Dyncast consists of the following Control-plane and Data-plane operations:

* Dyncast Control Plane:

  - Dyncast Service ID Notification: the D-SID, an anycast IP address, should be available and known. This can be achieved in different ways. For example, use a special range or coding of anycast IP address as service IDs or using the DNS.

- Dyncast Binding ID Notification: the mapping of (D-SID, D-BID), i.e., service ID and the binding address, should be notified to the D-Routers when the service instance starts (or stops). Various ways can be used, for example, EBGP or management system notification.

- Metrics Notification: D-MA have to be able to share the metrics for a service and its binding ID so that D-Routers can select the "best" instance for each new service demand.

* Dyncast Data Plane:

- New service demand: an ingress D-Router selects the most appropriate egress in terms of the network status and the computing resources of the instances of the requested service.

- Instance Affinity: Make sure the subsequent packets of an existing service demand are always delivered to the same service instance so that they can be served by the same service instance.

6. Summary

This draft introduces a Dyncast architecture that enables the service demand to be sent to an optimal service instance. It can dynamically adapt to the computing resources consumption and network status change. 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.

More discussion and input on control plane and data plane approach are welcome.

7. Security Considerations

The computing resource information changes over time very frequent with the creation and termination of service instance. When such information is carried in routing protocol, too many updates can make the network fluctuate. Control plane approach should take it into considerations.

More thorough security analysis to be provided in future revisions.

8. IANA Considerations

This document does not make any request to IANA.

9. Contributors

Huijuan Yao

yaohuijuan@chinamobile.com

China Mobile

Xia Chen

jescia.chenxia@huawei.com

Huawei

10. 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.liu-dyncast-ps-usecases] Liu, P., Eardley, P., Trossen, D., Boucadair, M., Contreras, L. M., and C. Li, "Dynamic-Anycast (Dyncast) Use Cases and Problem Statement", Work in Progress, Internet-Draft, draft-liu-dyncast-ps-usecases-03, 7 March 2022, <https://www.ietf.org/archive/id/draft-liu-dyncast-ps-usecases-03.txt>.

Acknowledgements

TBD

Authors' Addresses

Yizhou Li
Huawei Technologies
Email: liyizhou@huawei.com

Luigi Iannone
Huawei Technologies
Email: Luigi.iannone@huawei.com

Dirk Trossen
Huawei Technologies
Email: dirk.trossen@huawei.com

Peng Liu
China Mobile
Email: liupengyjy@chinamobile.com

Cheng Li
Huawei Technologies
Email: c.l@huawei.com