Workgroup: rtgwg
Internet-Draft:
draft-li-dyncast-architecture-08
Published: 17 January 2023
Intended Status: Standards Track
Expires: 21 July 2023
Authors: Y. Li
Huawei Technologies
D. Trossen
Huawei Technologies
C. Li, Ed.
Huawei Technologies
L. Iannone
Huawei Technologies
P. Liu
China Mobile

Dynamic-Anycast Architecture

Abstract

This document describes an architecture for the Dynamic-Anycast (Dyncast). It includes an architecture overview, main components, and the workflow of control plane and dataplane.

Status of This Memo

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

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

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

This Internet-Draft will expire on 21 July 2023.

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

Table of Contents

1. Introduction

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.

As described in [I-D.liu-can-ps-usecases],traffic steering that takes into account computing resource metrics would benefit several use-cases, such as AR/VR. This document provides an architectural framework, which will enable compute- and network-aware traffic steering decisions in edge computing.

The Dyncast architecture proposed in this document is an ingress-based overlay architecture for the selection of a suitable service instance from a set of possibly several ones, where 'suitable' may be determined through a combination of networking and computing related metrics.

Dyncast assumes that there are multiple service instances running on different edge nodes, globally providing 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 of dispatching service demands, by taking into account both service instances status as well as network state (e.g., paths length, price and their congestion).

This document describes an architecture to realize Dyncast, the workflow of main procedures in control and data plane.

2. Terminology

*Dyncast: As defined in [I-D.liu-can-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-can-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-can-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 maintain instance affinity, i.e., forwards packets belonging to the same service demand to the same instance.

*Ingress D-Router: a service access point for Dyncast clients. It makes the routing decision to encapsulate the service packets into an overlay path to an Egress D-Router linked to the most suitable edge site/instance.

*Egress D-Router: An Egress D-Router is the egress endpoint of an overlay path.

*D-MA: Dyncast Metric Agent (D-MA): A dyncast specific agent able to gather and send metric updates (from both network and instance perspective) but not performing forwarding decisions. May run on a D-Router, but it can be also implemented 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 instance.

2.1. 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. Architecture and Components

Dyncast assumes that there are multiple equivalent service instances running on different edge sites, 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 status. The architecture is shown below.

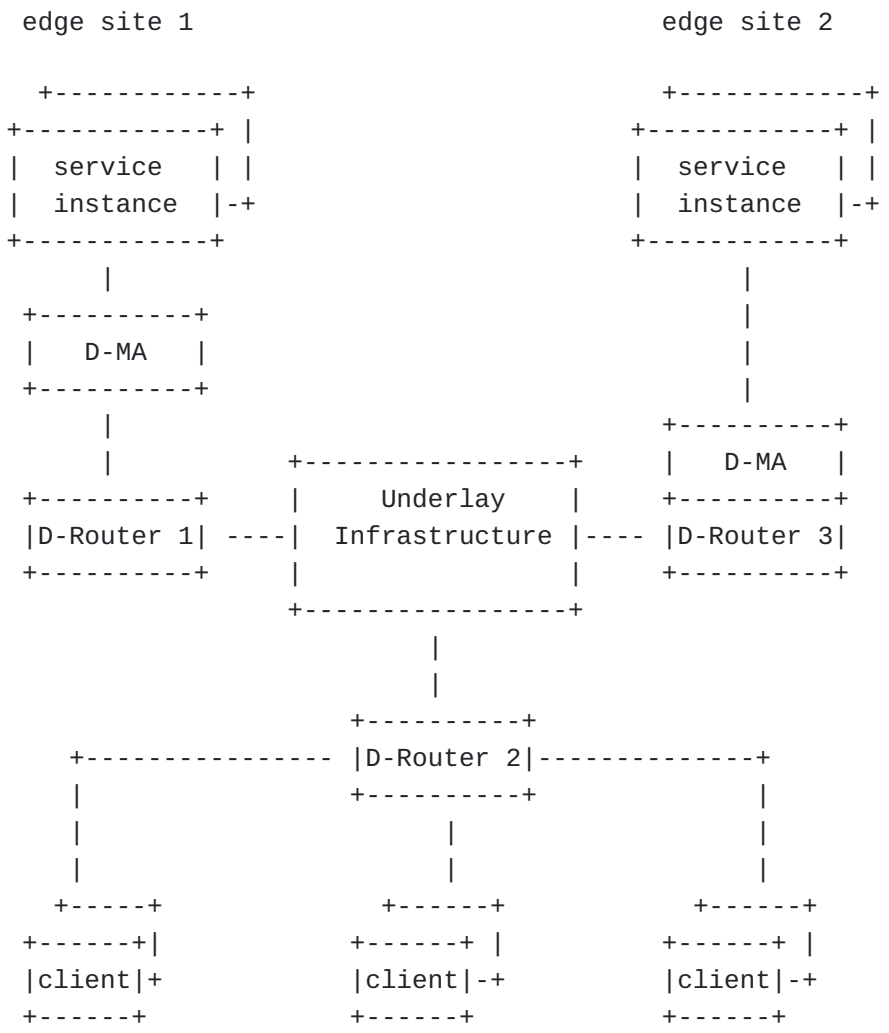

Figure 1: Dyncast Architecture.

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. The information of resource and status will be presented as one or more metrics and associate with the D-SID. D-MAs also need to send the D-SID routes with computing-related metrics to the Egress D-Routers proactively. A D-MA 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. In the case

of running in a D-Router, the D-SID routes with computing-related metrics can be synchronized up within the system.

The D-Router is the main element in a Dyncast network. There are two types of D-Routers: Egress D-Router and Ingress D-Router.

* Egress D-Router: An Egress D-Router is the egress endpoint of an overlay path. It distributes information on D-SID metrics, as well as information of identifying the Egress D-Router, to the associated Ingress D-Routers. Normally, a site may be linked to one or more Egress D-routers.

* Ingress D-Router: a service access point for Dyncast clients. It will receive the information on D-SIDs metrics and egress D-Routers as distributed from associated Egress D-Routers, and generate the routing decision based on the network-related and computing-related metrics. Some network metrics can be learned on the ingress D-Router by detecting, such as the latency and bandwidth from the ingress to an egress D-Router. Usually, the routing decision will indicate the Ingress D-Router to encapsulate the service packets into an overlay path to an Egress D-Router linked to the most suitable edge site/instance.

Note: Depending on deployment requirements, per-instance computing-related metrics or per-site aggregated computing-related metrics can be used in routing decision. In deployment, using per-site aggregated computing-related metrics is a more practical choice.

When a service packet is decapsulated on an Egress D-Router, it will be sent to the service instance within the edge site by looking up the destination address in the FIB.

Within the edge site, Load balancing and/or NAT may be used if needed. When using NAT, the D-SID will be translated into a unicast address associated to a specific service instance, and this unicast address is called D-BID(Dyncast Binding ID). There is no needed to configure a B-SID for a service instance if NAT is not needed.

In Figure 1, the "underlay infrastructure" indicates the general IP infrastructure that does not need to support Dyncast. The Dyncast routes will be distributed among the overlay D-Routers, and will not affect the underlay nodes. An implementation may use an IP protocol or an IT solution in Layer 3, Layer 3.5, Layer 4 or even Layer 7 to distribute the Dyncast routes with computing-related metrics. This document will not define a specific solution in current stage.

The above text describe a distributed architecture of Dyncast. However, a centralized architecture can also work. In the centralized architecture, the computing-related metrics from the Egress D-Routers or D-MAs are collected by a centralized controller/

PCE. Considering both the network-related metrics and computing-related metrics, the Controller/PCE can calculate the best path for a D-SID and send it to the related Ingress D-Router. An implementation may use an IP protocol or an IT solution in Layer 3, Layer 3.5, Layer 4 or even Layer 7 to collect the Dyncast routes with computing-related metrics. This document will not define a specific solution in current stage.

4. Dyncast Architecture Workflow

The following section provides an overview of how the Dyncast solution works in the distributed architecture.

4.1. Service Announcements

Normally, a service is associated with an IP address, which can be an IPv6 address. In Dyncast, this IP address is called D-SID, and it can be used by multiple service instances, which makes it as an Anycast address in routing. This address can be learned via DNS resolution or other mechanisms, this document will not limit this.

4.2. Metric Distributions

As described above, a D-MA will collect computing-related metrics and associates the metrics to a related D-SID route. It will also send the D-SID route with the metrics to the connected Egress D-Router. The Egress D-Router will distribute the D-SID route with the metrics to the related Ingress D-Routers. The metrics include the computing-related metrics and potential other network metrics if needed.

As explained in the problem statement document [I-D.liu-can-ps-usecases], computing metrics may change very frequently, when and how frequent such information should be distributed 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.

The following example is provided for better understanding of how Dyncast metrics are distributed. Routes of D-SID1 and D-SID2 with related metrics are sent from the D-MA to D-Router 2. D-Router 2 generates overlay routes of D-SID1 and D-SID2, and distribute them to the associated Ingress D-Router 1.

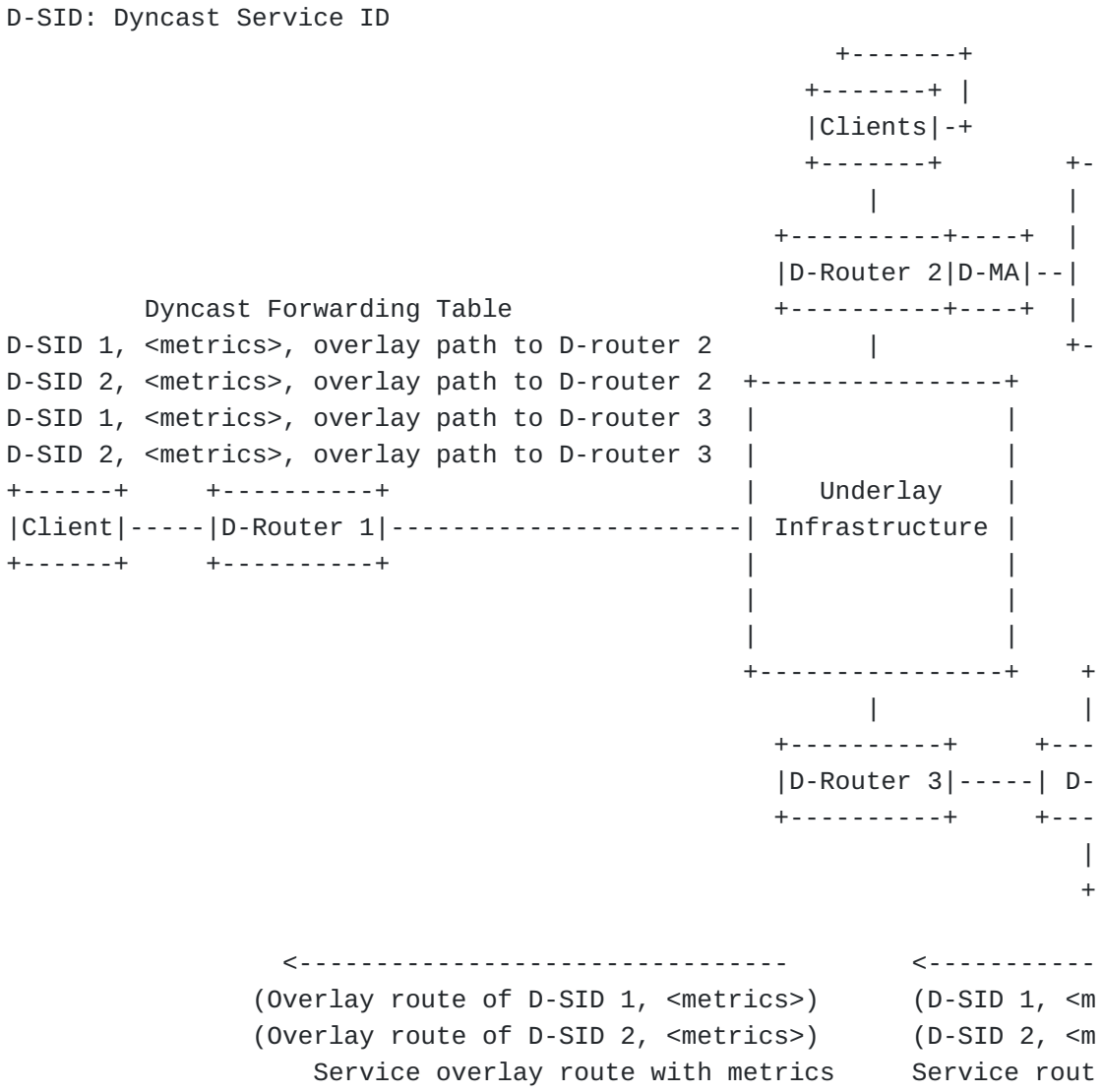

Figure 2: Dyncast deployment example.

4.3. Service Demand Handling

Ingress D-Routers can make their routing decision based on the received routes and metrics. The metrics include network-related metrics and computing-related metrics, while the network metrics can be learned by detecting on the ingress D-Router, and the computing-related metrics are learned from the received D-SID's routes. This document will not define any algorithm of making the routing decision. The routing decision may indicate the Ingress D-Router to encapsulate the service packets into an overlay path towards to the 'best' Egress D-Router. In the example provided in above figure, the Ingress D-Router 1 generate the routes of D-SID1 and D-SID2 with next hop as an overlay path to a specific Egress D-router 2 or D-Router 3.

When a new service transaction starts, an initial service packet is sent from the client to its Dyncast Ingress D-Router. Upon receiving the service packet, the ingress will forward the packets to the 'best' egress D-Router through an overlay path. When the service packet reaches the egress D-router, the outer header of the overlay encapsulation will be decapsulated and the inner service packet will be sent to the 'best' service instance.

4.4. Instance Affinity

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.

Note: 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.

This document will not define the specific mechanism of affinity, and it can be discussed in specific solution drafts.

5. 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.

6. IANA Considerations

TBD

7. Contributors

*Huijuan Yao, yaohuijuan@chinamobile.com, China Mobile

*Xia Chen, jescia.chenxia@huawei.com, Huawei

*Jianwei Mao, maojianwei@huawei.com

*Hang Shi, shihang9@huawei.com, Huawei

8. References

8.1. Normative References

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

8.2. Informative References

[I-D.liu-can-ps-usecases]
Liu, P., Eardley, P., Trossen, D., Boucadair, M., Contreras, L. M., Li, C., and Y. Li, "Computing-Aware Networking (CAN) Problem Statement and Use Cases", Work in Progress, Internet-Draft, draft-liu-can-ps-usecases-00, 23 October 2022, <https://www.ietf.org/archive/id/draft-liu-can-ps-usecases-00.txt>.

[I-D.liu-dyncast-ps-usecases]
Liu, P., Eardley, P., Trossen, D., Boucadair, M., Contreras, L. M., Li, C., and Y. Li, "Dynamic-Anycast (Dyncast) Problem Statement and Use Cases", Work in Progress, Internet-Draft, draft-liu-dyncast-ps-usecases-04, 8 July 2022, <https://www.ietf.org/archive/id/draft-liu-dyncast-ps-usecases-04.txt>.

Authors' Addresses

Yizhou Li
Huawei Technologies
China

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
China

Email: liupengyjy@chinamobile.com

Cheng Li (editor)
Huawei Technologies
China

Email: c.l@huawei.com