dyncast
Internet-Draft
Intended status: Informational
Expires: August 19, 2021

Y. Li
L. Iannone
D. Trossen
Huawei Technologies
P. Liu
China Mobile
February 15, 2021

Dynamic-Anycast Architecture
draft-li-dyncast-architecture-00

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

Copyright Notice

Table of Contents

1. Introduction

Edge computing is expanding from a 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, for example, 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.

Networking taking into account computing resource metrics seems to be an interesting ~~paradigm~~ approach that fits numbers of use-cases that would benefit from such capability [I-D.liu-dyncast-ps-usecases]. Yet, more investigation is still needed in key areas for this ~~paradigm~~approach and, to this end, this document aims at providing an architectural framework, which will enable service notification, status update, and service dispatch in edge computing~~.~~.

The Dyncast architecture presents an ~~anycast~~ anycast-based service and access model addressing the problematic aspects of existing network layer edge computing service deployments, including the unawareness of computing resource information of a 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.

Commenté [BMT1]: Do you mean « service function instances »?

Commenté [BMT2]: This is more like a service function chain. No?

Mis en forme : Surlignage

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 service represents a defined endpoint of functionality encoded according to the specification for said service.

Commenté [BMT3]: So, this is not dispatched in many nodes as suggested in the intro?

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

Commenté [BMT4]: This is a service function instance, then. No?

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~~maintain instance affinity, i.e., forwards packets belonging to the same service demand to the same service instance.

Mis en forme : Surlignage

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

Commenté [BMT5]: To where ?

D-Forwarder: An optional element able to forward packets towards a service instance, while not receiving any metric and as such not being able to make any decision when a new service demand arrives. it relies on a D-Router for the decision, it only guarantees instance affinity.

D-SID: Dyncast Service ID, an identifier representing a service, which the clients use to access the said service. Such identifier ~~identifies~~ is common to 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 instances through an anycast model, i.e., choosing one instance among all available instances.

D-BID: Dyncast Binding D-Node, 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~~ by a D-SID but with different ~~Dyncast Binding IDs~~D-BIDs.

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

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

3. Architecture Main Concepts

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~~may have a significant number of instances running on different edges as a function of local deployment. 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 gateways in edge data centers.

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-enabled router (named D-Router), while other ~~deployement~~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

Commenté [BMT6]: Should be better differentiated from a D-Router.

Commenté [BMT7]: I would avoid this acronym as it may be confused with Segment Identifier (SID) used in Segment Routing

Commenté [BMT8]: Is this is a name to reach a service or any ID to uniquely identify a service instance (e.g., 12254) where the semantic of the ID is local to the service provider?

Commenté [BMT9]: What is a D-Node?

Commenté [BMT10]: This means D-SID is explicated in the demand?

Commenté [BMT11]: Do you mean « forwarded »

Commenté [BMT12]: The identification is local. I guess this is an opaque value. Do you require any internal structure for this ID?

Commenté [BMT13]: This can be disseminated by the service instances or be retrieved on-demand by the D-MA using some "means" to be defined. Right?

Do you have in mind solutions such as https://tools.ietf.org/html/draft-chunduri-ospf-operator-defined-tlvs-02?

Commenté [BMT14]: That is ?

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 ~~achitectural~~architectural document does not make any specific assumption on this matter. This documents only assumes that:

Commenté [BMT15]: Please explicit what is meant here.

Commenté [BMT16]: That is as a function of the distribution criteria that are local to the service. Right?

- o 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.

Commenté [BMT17]: D-Router are service elements. The use of "router" may not be convenient here.

These nodes are more like Session Border Elements (SBEs) (and may be Data Border Elements (DBEs) as well)

- o 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'.

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

```
       edge site 1          edge site 2            edge site 3

        +------------+                          +------------+
      +------------+ |                        +------------+ |
      |  service   | |                        |  service   | |
      |  instance  |-+                        |  instance  |-+
      +------------+                          +------------+
            |                                        |
       +----------+                                  |
       |   D-MA   |                                  |
       +----------+                             +----------+
            |           +-----------------+     |   D-MA   |
       +----------+     |                 |     +----------+
       |D-Router 1| ----|  Infrastructure |---- |D-Router 3|
       +----------+     |                 |     +----------+
            |           +-----------------+          |
            |                    |                   |
            |                    |                   |
      +-----------+         +----------+             |
      |D-Forwarder|         |D-Router 2|             |
      +-----------+         +----------+             |
            |                    |                   |
            |                    |                   |
         +-----+              +------+          +------+
       +------+|            +------+ |        +------+ |
       |client|+            |client|-+        |client|-+
       +------+             +------+          +------+
```

Figure 1: Dyncast Architecture.

Figure 2 shows an example of Dyncast ~~deployement~~deployment, with 2 service ~~instatiated~~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. The edge site 3 uses a standalone D-MA to report its metrics to the Dyncast system and, since no client is present at that edge, there is no need of a D-Router. Edge site 2 instead, collocates the D-MA with a D-router since client are present.

Commenté [BMT18]: This figure is mixing organic nodes vs. functions.

Commenté [BMT19]: I would position D-Router as an overlay service node. This is more clean from an architectural standpoint.

```
    D-SID: Dyncast Service ID
    D-BID: Dyncast Binding ID

            Service/Metrics Information
            (D-SID 1, D-BID 21, <metrics>)
            (D-SID 2, D-BID 22, <metrics>)
           <----------------->

                                 +--------+
                               +--------+ |           D-SID 1
                               |Clients|-+          +---------+
                               +--------+       +--|D-BID 21| instance 1
                                   |            |  +---------+
                             +----------+----+  |              Edge 2
                             |D-Router 2|D-MA|--|     D-SID 2
                             +----------+----+  |  +---------+
                                   |            +--|D-BID 22| instance 2
                           +----------------+      +---------+
                           |                |
                           |                |
   +-------+ +-----------+ |                |
   |Client|--|D-Router 1|--| Infrastructure |
   +-------+ +-----------+ |                |
                           |                |     D-SID 2
                           |                |  +---------+
                           +----------------+ +---|D-BID 32| instance 3
                                   |           |   +---------+
                                   |       +-------+           Edge 3
                                   +-------| D-MA |
                                           +-------+  D-SID 1
                                               |   +---------+
                                               +---|D-BID 31| instance 4
                                                   +---------+

           <---------------------------------->
              (D-SID 2, D-BID 32, <metrics>)
              (D-SID 1, D-BID 31, <metrics>)
               Service/Metrics Information
```

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,

Commenté [BMT20]: Why Clients have to be aware of this?

SID are IMO opaque value that is internal to the service. Of course, the information can be "leaked" during service resolution or service attachment, but the client does not need to perform specific behavior for that SID. No?

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/deletetd 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 ~~relavant~~relevant information has to be collected by D-MA associated ~~to~~ for 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.

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:

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

o (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

Commenté [BMT21]: Indeed.

For such case, I guess a service instance in Figure 2 can't be a DNS service instance.

Mis en forme : Surlignage

Commenté [BMT22]: You may refer to the metrics mentioned in the figure.

Commenté [BMT23]: You may remind that this functionality is supported by load balancers to make sure that the same instance is used for the same flow.

Commenté [BMT24]: Unavailable, in general.

Commenté [BMT25]: I guess this a form of deregistering a service instance.

the two instances of the two services running on edge 3. These information is:

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

o (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~~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 and discover relevant information.

Commenté [BMT26]: Which is a service overlay. This is worth to be reflected in the architecture.

Commenté [BMT27]: In order to avoid the dependency on the underlay network, it is better to cover this as part of the server overlay. Dedicated tools can be enabled between D-Routers without requiring a change to the underlay network.

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 instances 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-Node 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, ~~threshhold~~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.

In the Dyncast ~~Architecture~~ architecture there are two possible type of ingress, namely D-Routers and D-Forwarders, which are discussed in the following.

4.2.1. Service Demand Dispatch and Instance Affinity on D-Routers ingress

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

Commenté [BMT28]: For QUIC, we may rely upon the destination Connection Identifier (that can be coordinated among several service instances).

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

4.2.2. Service Demand Dispatch and Instance Affinity on D-Forwarders ingress

When a D-Router maintains the binding table, the memory consumed is determined by the number of different service demands that a Dyncast 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, creating a concern regarding the memory space consumption for the binding table at the Dyncast ingress node. To alleviate that concern, the Dyncast Forwarder (D-Forwarder for short) can be used and take an active role.

The D-Forwarder is deployed closer to the clients and it normally handles the traffic and service demands of a single or a few clients. In this case, the memory required by the binding table will be much smaller since the number of entries is now limited to the number of local clients only. Furthermore, the D-Forwarder is not a D-Router, that is to say, it does not participate in the status update about network and computing metrics among D-Routers. A D-Forwarder does not determine the best egress to forward packets when there is a new service demand. Instead, it has to learn such information from a D-Router and maintains it to ensure the instance affinity for subsequent packets. In this way, the D-routers may be relieved from binding table maintenance.

Figure 4 shows the interaction between D-Forwarders and D-Routers. The figures show a scenario similar to Figure 2, with the addition of a D-Forwarder in front of D-Router1. When a new service demand arrives at a D-Forwarder, the latter has no suitable entry in its binding table that allows forwarding the packet to an egress. As a consequence, the D-Forwarder forwards the service demand to a D-Router, while marking the 'miss' of matching the demand onto a suitable binding address in the forwarded packet. Upon receiving the service demand, the D-Router, having access to all of the relevant metric information, will select the most suitable egress, i.e., service instance, and forward the packet as a service request to the chosen service instance. Based on the 'miss' indication in the received service demand, the D-Router will also inform the D-Forwarder about the selected egress. This will allow the D-Forwarder to maintain the binding table to ensure the mapping of any subsequent service demand.

The control messages exchange between the D-Forwarder and its corresponding D-Router needs to be defined, but is out of the scope of this document. D-Routers have to be also able to inform D-Forwarders if there is any issue concerning packet delivery. For instance, an ingress D-Router may find out that the traffic from the D-Forwarder is going to an unreachable egress, e.g., due to node failure. In such a case, it should inform the D-Forwarder about the issue as soon as possible. The information exchange may also contain possible countermeasures.

```
    D-SID: Dyncast Service ID
   D-BID: Dyncast Binding ID
                                         +-------+
                                       +-------+ |           D-SID 1
                                       |Clients|-+         +--------+
                                       +-------+        +--|D-BID 21|
                                           |            |  +--------+
                                     +----------+----+  |
                                     |D-Router 2|D-MA|--|    D-SID 2
   +------+                          +----------+----+  |  +--------+
   |Client|                                |            +--|D-BID 22|
   +------+                        +----------------+      +--------+
      |    Service Demand          |                |
      |   (Flow X, D-SID 2)        |                |
      |   --------------->         |                |
   +-----------+     +----------+  |                |
   |D-Forwarder|-----|D-Router 1|--| Infrastructure |
   +-----------+     +----------+  |                |
      |   <---------------         |                |
      |(Flow X, D-SID 2, D-BID 32) |                |       D-SID 2
      |     Binding Info           |                |      +--------+
   +------+                        +----------------+  +---|D-BID 32|
   |Client|                                |           |   +--------+
   +------+                                |       +------+
                                           +-------| D-MA |
                                                   +------+  D-SID 1
                                                       |   +--------+
                                                       +---|D-BID 31|
                                                           +--------+
```

Figure 4: Service Demand in presence of a D-Forwarder

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.

  * Mapping Update Notification: D-Router notifies D-Forwarder of incoming service demand of mapping from service ID to binding IP according to the local metric information. This notification is sent upon receiving a service demand (from D-Forwarder) with 'miss' indication.

- 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. An ingress D-Forwarder selects the binding address information for the received service ID, if available. Otherwise, the service demand is forwarded with 'miss' indication set.

  * 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. References

9.1. Informative References

[I-D.liu-dyncast-ps-usecases]
Liu, P., Willis, P., and D. Trossen, "Dynamic-Anycast (Dyncast) Use Cases and Problem statement", draft-li-dyncast-ps-usecases-01 (work in progress), February 2021.

~~9.2. 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>.

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