rtgwg
Internet-Draft
Intended status: Standards Track
Expires: 21 July 2023

Y. Li
L. Iannone
D. Trossen
Huawei Technologies
P. Liu
China Mobile
C. Li, Ed.
Huawei Technologies
17 January 2023

Dynamic-Anycast Architecture
draft-li-dyncast-architecture-08

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.

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.

```
      edge site 1                                edge site 2

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

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.

```
 D-SID: Dyncast Service ID
                                                          +-------+
                                                        +-------+ |
                                                        |Clients|-+         +-------
+
                                                        +-------+        +--|D-SID 1
| instance 1
                                                            |            |  +-------
+
                                                      +----------+----+  |
    Edge 2
                                                      |D-Router 2|D-MA|--|
          Dyncast Forwarding Table                    +----------+----+  |  +-------
+
 D-SID 1, <metrics>, overlay path to D-router 2           |            +--|D-SID 2
| instance 2
 D-SID 2, <metrics>, overlay path to D-router 2  +----------------+      +-------
+
 D-SID 1, <metrics>, overlay path to D-router 3  |                |
 D-SID 2, <metrics>, overlay path to D-router 3  |                |
 +------+     +----------+                       |   Underlay     |
 |Client|-----|D-Router 1|-----------------------| Infrastructure |
 +------+     +----------+                       |                |
                                                 |                |
                                                 |                |        +-----
--+
                                                +----------------+    +---|D-SID
 2| instance 3
                                                         |             |   +-----
--+
                                                      +----------+     +------+
       Edge 3
                                                      |D-Router 3|-----| D-MA |
                                                      +----------+     +------+
                                                                          |   +-----
--+
                                                                          +---|D-SID
 1| instance 4
                                                                              +-----
--+
                  <--------------------------------        <--------------
                 (Overlay route of D-SID 1, <metrics>)      (D-SID 1, <metrics>)
                 (Overlay route of D-SID 2, <metrics>)      (D-SID 2, <metrics>)
                    Service overlay route with metrics     Service route with me
trics
```

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

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

P. Liu
China Mobile
P. Eardley
BT
D. Trossen
Huawei Technologies
M. Boucadair
Orange
LM. Contreras
Telefonica
C. Li
Y. Li
Huawei Technologies
8 July 2022

Dynamic-Anycast (Dyncast) Problem Statement and Use Cases
draft-liu-dyncast-ps-usecases-04

Abstract

Many service providers have been exploring distributed computing techniques to achieve better service response time and optimized energy consumption. Such techniques rely upon the distribution of computing services and capabilities over many locations in the network, such as its edge, the metro region, virtualized central office, and other locations. In such a distributed computing environment, providing services by utilizing computing resources hosted in various computing facilities (e.g., edges) is being considered, e.g., for computationally intensive and delay sensitive services. Ideally, services should be computationally balanced using service-specific metrics instead of simply dispatching the service requests in a static way or optimizing solely connectivity metrics. For example, systematically directing end user-originated service requests to the geographically closest edge or some small computing units may lead to an unbalanced usage of computing resources, which may then degrade both the user experience and the overall service performance. We have named this kind of network with dynamic sharing of edge compute resources "Computing-Aware Networking" (CAN).

This document provides the problem statement and the typical scenarios of CAN, which is to provide the service equivalency by steering traffic dynamically to the appropriate service instance based on the basic edge computing deployment.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Network and Computing convergence has been evolving in the Internet for considerable time. With Content Delivery Networks (CDNs) 'frontloading' access to many services, over-the-top service provisioning has become a driving force for many services, such as video, storage and many others. In addition, network operators have extended their capabilities by complementing their network infrastructure by developing CDN capabilities, particularly in edge sites. Compared to a CDN-based content cache capability, more diverse computing resource need to be provided for general edge computing in an on-demand manner.

The reason of the fast development of this converged network/compute infrastructure is user demand. On the one hand, users want the best experience, e.g., expressed in low latency and high reliability, for new emerging applications such as high-definition video, AR and VR, live broadcast and so on. On the other hand, users want the stable experience when moving to different areas.

Generally, edge computing aims to provide better response times and transfer rates compared to Cloud Computing, by moving the computing towards the edge of a network. Edge computing can be built on embedded systems, gateways, and others, all being located close to end users' premises. 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 behaving as service nodes.

That brings about the key problem of deploying and scheduling traffic to the most suitable computing resource in order to meet the users' (service-specific) demand.

Depending on the location of an edge and its capacity, different computing resources can be contributed by each edge to deliver a service. At peak hours, computing resources attached to a client's closest edge may not be sufficient to handle all the incoming service requests. Longer response times or even dropping of requests can be experienced by users. Increasing the computing resources hosted on each edge to the potential maximum capacity is neither feasible nor economically viable in many cases. Offloading computation intensive processing to the User devices would give the huge pressure of battery, and the needed data set (for the computation) that may not exist on the user device because of the size of data pool or due to data governance reasons.

While service providers often have their own sites, which in turn have been upgraded to the edge sites, a specific service should be deployed in multiple edge sites to meet the users' demand. However, only the deployment itself might not enough to fully guarantee the quality of service. Instead, functional equivalency must be ensured by deploying instances for the same service across edge sites for better availability. Furthermore, load is to be kept balanced for both static and dynamic scenarios. For this, traffic needs to be dynamically steered to the "best" service instance. For this, traffic must be delivered to optimal edge sites according to information that may need to include, e.g., computing information, where the notion of 'best' may highly depend on the application demand.

A particular example is the popular and pervasive 5G MEC service. In 5G MEC, ULCL UPFs are deployed close to edge sites, which are capable of effectively classifying & switching uplink traffic to the suitable computing-resources that might be located either in local-area DNs, operators' DNs, or even 3rd-party's DNs. Thru possibly using some 'intelligent' criteria, this could warrant the selection of resources with either low-latency, high-throughput, high-computational power or all-involved requirements.

This document describes sample usage scenarios as well as key areas in which current solutions lead to problems that ultimately affect the deployment (including the performance) of edge services, and proposes the desired features of the CAN system. Those key areas target the identification of candidate solution components.

2. Definition of Terms

This document makes use of the following terms:

CAN: Aiming at computing and network resource optimization by steering traffic to appropriate computing resources considering not only routing metric but also computing resource metric and service affiliation.

Service: 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: 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.

Service identifier: Used to uniquely identify a service, at the same

time identifying the whole set of service instances that each represent the same service behavior, no matter where those service instances are running.

3. Problem Statement

3.1. Multi-deployment of Edge Sites and Service

Since edge computing aims at a closer computing service based on the shorter network path, there will be more than one edge sites with the same application in the city/province/state, a number of representative cities have deployed multi-edge sites and the typical applications, and our hypothetical model is based on that as figure 1 shows. Moreover, we assumes that the service deployment and service discovery has been done based on the hypothetical model, so the main problem is to steer traffic among multiple edge sites.

Traffic needs to be steered to the appropriate edge sites to ensure the application demands, because in some cases, the 'closest' is not the 'best', there will be the variable statues of computing and network could be summarized as:

o Closest site may not have enough resource, the load may dynamically change.

o Closest site may not have related resource, heterogeneous hardware in different sites.

Therefore, more enhancement based on edge computing is need. Because for edge computing, the service request always be steered to the closest edge site. Some existing methods allow to balance the load of edge sites but might not be enough to consider single factors, which will be described in Section 4.

We assume that clients access one or more services with an objective to meet a desired user experience. Each participating service may be realized at one or more places in the network (called, service instances). Such service instances are instantiated and deployed as part of the overall service deployment process, e.g., using existing orchestration frameworks, within so-called edge sites, which in turn are reachable through a network infrastructure via an egress router.

When a client issues a service request to a required service, the request is being steered by its ingress router to one of the available service instances that realize the requested service. Each service instance may act as a client towards another service, thereby seeing its own outbound traffic steered to a suitable service instance of the request service and so on, achieving service composition and chaining as a result.

The aforementioned selection of one of candidate service instances is done using traffic steering methods , where the steering decision may take into account pre-planned policies (assignment of certain clients to certain service instances), realize shortest-path to the 'closest' service instance, or utilize more complex and possibly dynamic metric information, such as load of service instances, latencies experienced or similar, for a more dynamic selection of a suitable service instance.

It is important to note that clients may move throughout the execution of a service, which may, as a result, position other service instance 'better' in terms of latency, load, or other metrics. This creates a (physical) dynamicity that will need to be catered for.

As a summary, Figure 1 outlines the main aspects of the assumed system model for realizing the use cases that follow next.

```
       +------------+      +------------+       +------------+
      +------------+ |    +------------+ |     +------------+ |
      |    Edge    | |    |    Edge    | |     |    Edge    | |
      |   Site 1   |-+    |   Site 2   |-+     |   Site 3   |-+
      +-----+------+      +------+-----+       +------+-----+
            |                   |                    |
       +----+-----+        +-----+----+         +-----+----+
       |  Egress  |        |  Egress  |         |  Egress  |
       | Router 1 |        | Router 2 |         | Router 3 |
       +----+-----+        +-----+----+         +-----+----+
            |                   |                    |
            |           +--------+--------+           |
            |           |                 |           |
            +-----------|  Infrastructure |-----------+
                        |                 |
                        +--------+--------+
                                 |
                            +----+----+
                            | Ingress |
            +---------------|  Router |--------------+
            |               +----+----+              |
            |                    |                   |
         +--+--+              +--+---+           +---+--+
        +------+|            +------+ |         +------+ |
        |Client|+            |Client|-+         |Client|-+
        +------+             +------+           +------+
```

Figure 1: CAN Use Case Model

3.2. Traffic Steering among Edges Sites

This section shows the necessity of traffic steering among different edges in the real city, considering the mobility of the people in different time slot, events, etc.

Figure 2 shows a common way to deploy edge sites in the metro. There is an edge data center for metro area which has high computing resource and provides the service to more UEs at the working time. Because more office buildings are in the Metro area. And there are also some remote edge sites which have limited computing resource and provide the service to the UEs closed to them.

The application such as the AR/VR, video recognition could be deployed in both the edge data center in metro area and the remote edge sites. In this case, the service request and the resource are matched well. Some potential traffic steering may needed just for special service request or some small scheduling demand.

```
      +----------------+    +---+                  +------------+
    +----------------+ |- - |UE1|                +------------+ |
    | +------------+ | |    +---+              +--|    Edge    | |
    | |Edge server| | |    +---+        +- - -|PE|            | |
    | +------------+ | |- - |UE2|        |     +--|   Site 1   |-+
    | +------------+ | |    +---+                +------------+
    | |Edge server| | |     ...         |            |
    | +------------+ | +--+         Potencial      +---+ +---+
    | +------------+ | |PE|- - - - - - -+          |UEa| |UEb|
    | |Edge server| | +--+         Steering       +---+ +---+
    | +------------+ | |    +---+        |                 |
    | +------------+ | |- - |UE3|                +------------+
    | |  ... ...  | | |    +---+        |       +------------+ |
    | +------------+ | |     ...               +--|    Edge    | |
    |                | |    +---+        +- - -|PE|            | |
    |Edge data center|-+- - |UEn|              +--|   Site 2   |-+
    +----------------+      +---+                +------------+
    High computing resource              Limited computing resource
    and more UE at Metro area            and less UE at Remote area
```

Figure 2: Common Deployment of Edge Sites

Figure 3 shows that when it goes to non working time, for example at weekend or daily night, more UEs move to the remote area that are close to their house or for some weekend events. So there will be more service request at remote but with limited computing resource, while the rich computing resource might not be used with less UE in the Metro Area. It is possible for so many people request the AR/VR service at remote are but with the limited computing resource, moreover, as the people move from the metro area to the remote are, the edge sites served the common service such as intelligent transportation will also change, so it need to steer some traffic back to Metro center.

```
        +----------------+                          +------------+
      +----------------+ |                        +------------+ |
      | +------------+ | |  Steering traffic    +--|    Edge    | |
      | |Edge server| | |           +-----------|PE|            | |
      | +------------+ | |     +---+ |           +--|   Site 1   |-+
      | +------------+ | |- - |UEa| |    +----+----+-+----------+
      | |Edge server| | |    +---+ |    |           |           |
      | +------------+ | +--+       |  +---+ +---+ +---+ +---+ +---+
      | +------------+ | |PE|-------+  |UE1| |UE2| |UE3| |...| |UEn|
      | |Edge server| | +--+       |  +---+ +---+ +---+ +---+ +---+
      | +------------+ | |    +---+ |          |           |
      | +------------+ | |- - |UEb| |          +-----+-----+------+
      | |  ... ...  | | |    +---+ |              +------------+ |
      | +------------+ | |          |            +--|    Edge    | |
      |                | |          +-----------|PE|            | |
      |Edge data center|-+  Steering traffic    +--|   Site 2   |-+
      +----------------+                          +------------+
      High computing resource              Limited computing resource
      and less UE at Metro area            and more UE at Remote area
```

Figure 3: Steering Traffic among Edge Sites

There will also be the common variable of network and computing resources, for someone who is not moving but get a poor latency sometime. Because of other UEs' moving, a large number of request for temporary events such as vocal concert, shopping festival and so on, and there will also be the normal change of the network and computing resource status. So for some fixed UEs, it is also expected to steer the traffic to appropriate sites dynamiclly.

Those problems indicate that traffic needs to be steered among different edge sites, because of the mobility of the UE and the common variable of network and computing resources. Moreover, some apps in the following Section require both low latency and high computing resource usage or specific computing HW capabilities (such as local GPU); hence joint optimization of network and computing resource is needed to guarantee the QoE.

4. Use Cases

This section presents a non-exhaustive list of scenarios which require multiple edge sites to interconnect and to coordinate at the network layer to meet the service demands and ensure better user experience.

4.1. Computing-Aware AR or VR

Cloud VR/AR services are used in some exhibitions, scenic spots, and celebration ceremonies. In the future, they might be used in more applications, such as industrial internet, medical industry, and meta verse.

Cloud VR/AR introduces the concept of cloud computing to the rendering of audiovisual assets in such applications. Here, the edge cloud helps encode/decode and render content. The end device usually only uploads posture or control information to the edge and then VR/ AR contents are rendered in the edge cloud. The video and audio outputs generated from the edge cloud are encoded, compressed, and transmitted back to the end device or further transmitted to central data center via high bandwidth networks.

Edge sites may use CPU or GPU for encode/decode. GPU usually has better performance but CPU is simpler and more straightforward to use as well as possibly more widespread in deployment. Available remaining resources determines if a service instance can be started. The instance's 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 of quality of audio/ video and input command response times.

A Cloud VR service, such as a mobile gaming service, brings challenging requirements to both network and computing so that the edge node to serve a service request has to be carefully selected to make sure it has sufficient computing resource and good network path. 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, 20ms for motion-to-photon latency, packet loss rate is 2.4E-5; the typical computing requirements are 8K H.265 real-time decoding, 2K H.264 real-time encoding. We can further divide the 20ms latency budget into:

(i) sensor sampling delay(client), which is considered imperceptible by users is less than 1.5ms including an extra 0.5ms for digitalization and end device processing.

(ii) display refresh delay(client), which take 7.9ms based on the 144Hz display refreshing rate and 1ms extra delay to light up.

(iii) image/frame rendering delay(server), which could be reduced to 5.5ms.

(iv) network delay(network), which should be bounded to 20-1.5-5.5-7.9 = 5.1ms.

So the the budgets for server(computing) delay and network delay are almost equivalent, which make sense to consider both of the delay for computing and network. And it can't meet the total delay requirements or find the best choice by either optimize the network or computing resource.

Based on the analysis, here are some further assumption as figure 4 shows, the client could request any service instance among 3 edge sites. The delay of client could be same, and the differences of differente edge sites and corresponding network path has different delays:

o Edge site 1: The computing delay=4ms based on a light load, and the corresponding network delay=9ms based on a heavy traffic.

o Edge site 2: The computing delay=10ms based on a heavy load, and the corresponding network delay=4ms based on a light traffic.

o Edge site 3: The edge site 3's computing delay=5ms based on a normal load, and the corresponding network delay=5ms based on a normal traffic.

In this case, we can't get a optimal network and computing total delay if choose the resource only based on either of computing or network status:

o If choosing the edge site based on the best computing delay it will be the edge site 1, the E2E delay=22.4ms.

o If choosing the edge site based on the best network delay it will be the edge site 2, the E2E delay=23.4ms.

o If choosing the edge site based on both of the status it will be the edge site 3, the E2E delay=19.4ms.

So, the best choice to ensure the E2E delay is edge site 3, which is 19.4ms and is less than 20ms. The differences of the E2E delay is only 3~4ms among the three, but some of them will meet the application demand while some doesn't.

The conclusion is that it requires to dynamically steer traffic to the appropriate edge to meet the E2E delay requirements considering both network and computing resource status. Moreover, the computing resources have a big difference in different edges, and the 'closest site' may be good for latency but lacks GPU support and should therefore not be chosen.

```
     Light Load          Heavy Load           Normal load
   +------------+      +------------+       +------------+
   |    Edge    |      |    Edge    |       |    Edge    |
   |   Site 1   |      |   Site 2   |       |   Site 3   |
   +-----+------+      +------+-----+       +------+-----+
computing|delay(4ms)          |           computing|delay(5ms)
         |            computing|delay(10ms)         |
    +----+-----+        +-----+----+         +-----+----+
    |  Egress  |        |  Egress  |         |  Egress  |
    | Router 1 |        | Router 2 |         | Router 3 |
    +----+-----+        +-----+----+         +-----+----+
  newtork|delay(9ms)   newtork|delay(4ms)   newtork|delay(5ms)
         |                    |                    |
         |            +--------+--------+           |
         +-----------|  Infrastructure |-----------+
                      +--------+--------+
                               |
                          +----+----+
                          | Ingress |
         +---------------|  Router |--------------+
         |                +----+----+              |
         |                     |                   |
      +--+--+              +--+---+           +---+--+
    +------+|            +------+ |         +------+ |
    |Client|+            |Client|-+         |Client|-+
    +------+             +------+           +------+
                   clien delay=1.5+7.9=9.4ms
```

Figure 4: Computing-Aware AR or VR

Furthermore, specific techniques may be employed to divide the overall rendering into base assets that are common across a number of clients participating in the service, while the client-specific input data is being utilized to render additional assets. When being delivered to the client, those two assets are being combined into the overall content being consumed by the client. The requirements for sending the client input data as well as the requests for the base assets may be different in terms of which service instances may serve the request, where base assets may be served from any nearby service instance (since those base assets may be served without requiring cross-request state being maintained), while the client-specific

input data is being processed by a stateful service instance that changes, if at all, only slowly over time due to the stickiness of the service that is being created by the client-specific data. Other splits of rendering and input tasks can be found in[TR22.874] for further reading.

When it comes to the service instances themselves, those may be instantiated on-demand, e.g., driven by network or client demand metrics, while resources may also be released, e.g., after an idle timeout, to free up resources for other services. Depending on the utilized node technologies, the lifetime of such "function as a service" may range from many minutes down to millisecond scale. Therefore computing resources across participating edges exhibit a distributed (in terms of locations) as well as dynamic (in terms of resource availability) nature. In order to achieve a satisfying service quality to end users, a service request will need to be sent to and served by an edge with sufficient computing resource and a good network path.

4.2. Computing-Aware Intelligent Transportation

For the convenience of transportation, more video capture devices are required to be deployed as urban infrastructure, and the better video quality is also required to facilitate the content analysis. So, the transmission capacity of the network will need to be further increased, and the collected video data needs to be further processed, such as for pedestrian face recognition, vehicle moving track recognition, and prediction. This, in turn, also impacts the requirements for the video processing capacity of computing nodes.

In auxiliary driving scenarios, to help overcome the non-line-of-sight problem due to blind spot or obstacles, the edge node can collect comprehensive road and traffic information around the vehicle location and perform data processing, and then vehicles with high security risk can be warned accordingly, improving driving safety in complicated road conditions, like at intersections. This scenario is also called "Electronic Horizon", as explained in[HORITA]. For instance, video image information captured by, e.g., an in-car, camera is transmitted to the nearest edge node for processing. The notion of sending the request to the "nearest" edge node is important for being able to collate the video information of "nearby" cars, using, for instance, relative location information. Furthermore, data privacy may lead to the requirement to process the data as close to the source as possible to limit data spread across too many network components in the network.

Nevertheless, load at specific "closest" nodes may greatly vary, leading to the possibility for the closest edge node becoming overloaded, leading to a higher response time and therefore a delay in responding to the auxiliary driving request with the possibility of traffic delays or even traffic accidents occurring as a result. 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.

In video recognition scenarios, when the number of waiting people and vehicles increases, more computing resources are needed to process the video content. For rush hour traffic congestion and weekend personnel flow from the edge of a city to the city center, efficient network and computing capacity scheduling is also required. Those would cause the overload of the nearest edge sites if there is no extra method used, and some of the service request flow might be steered to others edge site except the nearest one.

4.3. Computing-Aware Digital Twin

A number of industry associations, such as the Industrial Digital Twin Association or the Digital Twin Consortium (https://www.digitaltwinconsortium.org/), have been founded to promote the concept of the Digital Twin (DT) for a number of use case areas, such as smart cities, transportation, industrial control, among others. The core concept of the DT is the "administrative shell" [Industry4.0], which serves as a digital representation of the information and technical functionality pertaining to the "assets" (such as an industrial machinery, a transportation vehicle, an object in a smart city or others) that is intended to be managed, controlled, and actuated.

As an example for industrial control, the programmable logic controller (PLC) may be virtualized and the functionality aggregated across a number of physical assets into a single administrative shell for the purpose of managing those assets. PLCs may be virtualized in order to move the PLC capabilities from the physical assets to the edge cloud. Several PLC instances may exist to enable load balancing and fail-over capabilities, while also enabling physical mobility of the asset and the connection to a suitable "nearby" PLC instance. With this, traffic dynamicity may be similar to that observed in the connected car scenario in the previous sub-section. Crucial here is high availability and bounded latency since a failure of the (overall) PLC functionality may lead to a production line stop, while boundary violations of the latency may lead to loosing synchronization with other processes and, ultimately, to production faults, tool failures or similar.

Particular attention in Digital Twin scenarios is given to the problem of data storage. Here, decentralization, not only driven by the scenario (such as outlined in the connected car scenario for cases of localized reasoning over data originating from driving vehicles) but also through proposed platform solutions, such as those in [GAIA-X], plays an important role. With decentralization, endpoint relations between client and (storage) service instances may frequently change as a result.

5. Conclusion

This document presents the problem statement and use cases of CAN in which we observe the demand for considering the dynamic nature of service requests in terms of requirements on the resources fulfilling them in the form of service instances. In addition, those very service instances may themselves be dynamic in availability and status, e.g., in terms of load or experienced latency.

As a consequence, we can get two obvious conclusion. One is that the traffic needs to be steered among different edge sites, another is that when steering traffic, the real-time network and computing resource status should be considered at the same time in an effective way. The problem of satisfying service-specific metrics to allow for selecting the most suitable service instance among the pool of instances available to the service throughout the network is a challenge.

6. Security Considerations

TBD.

7. IANA Considerations

TBD.

8. Contributors

The following people have substantially contributed to this document:

Peter Willis
BT

Tianji Jiang
China Mobile
tianjijiang@chinamobile.com

Markus Amend
Deutsche Telekom
Markus.Amend@telekom.de

9. Informative References

[RFC4786] Abley, J. and K. Lindqvist, "Operation of Anycast Services", BCP 126, RFC 4786, DOI 10.17487/RFC4786, December 2006, <https://www.rfc-editor.org/info/rfc4786>.

[TR22.874] 3GPP, "Study on traffic characteristics and performance requirements for AI/ML model transfer in 5GS (Release 18)", 2021.

[TR-466] BBF, "TR-466 Metro Compute Networking: Use Cases and High Level Requirements", 2021.

[HORITA] Horita, Y., "Extended electronic horizon for automated driving", Proceedings of 14th International Conference on ITS Telecommunications (ITST)", 2015.

[Industry4.0] Industry4.0, "Details of the Asset Administration Shell, Part 1 & Part 2", 2020.

[GAIA-X] Gaia-X, ""GAIA-X: A Federated Data Infrastructure for Europe"", 2021.

[MEC] ETSI, ""Multi-Access Edge Computing (MEC)"", 2021.

Acknowledgements

The author would like to thank Luigi IANNONE, Christian Jacquenet, Kehan Yao and Yuexia Fu for their valuable suggestions to this document.

Authors' Addresses

Peng Liu
China Mobile
Email: liupengyjy@chinamobile.com

Philip Eardley
BT
Email: philip.eardley@bt.com

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

Mohamed Boucadair
Orange
Email: mohamed.boucadair@orange.com

Luis M. Contreras
Telefonica
Email: luismiguel.contrerasmurillo@telefonica.com

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

Yizhou Li
Huawei Technologies
Email: liyizhou@huawei.com

rtgwg
Internet-Draft
Intended status: Informational
Expires: 8 September 2022

P. Liu
T. Jiang
China Mobile
P. Eardley
British Telecom
D. Trossen
C. Li
Huawei Technologies
7 March 2022

Dynamic-Anycast (Dyncast) Requirements
draft-liu-dyncast-reqs-02

Abstract

This draft provides requirements for an architecture addressing the problems outlined in the use case and problem statement draft for Dyncast[I-D.liu-dyncast-ps-usecases] .

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Computing service instances instantiated at multiple geographical edge sites are used to better realize an edge computing service in edge computing use cases, as shown in[I-D.liu-dyncast-ps-usecases]. To optimally deliver the service request to the most appropriate service instance is the fundamental requirement in such deployments. As shown in [I-D.liu-dyncast-ps-usecases], choosing the most appropriate service instance should take both, the computing resources available and the network path quality, into consideration. "Optimal" here additionally means the architecture and overall mechanism should be efficient, support high dynamism, while maintaining instance affinity, as shown in [I-D.liu-dyncast-ps-usecases].

This draft provides the requirements to realize the potential dynamic anycast architecture by alleviating the problems of existing solutions outlined in [I-D.liu-dyncast-ps-usecases]

2. Definition of Terms

Service: 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: 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.

Service identifier: Used to uniquely identify a service, at the same time identifying the whole set of service instances that each represent the same service behaviour, no matter where those service instances are running.

Anycast: An addressing and packet sending methodology that assign an "anycast" identifier for one or more service instances to which requests to an "anycast" identifier could be routed, following the definition in [RFC4786] as anycast being "the practice of making a particular Service Address available in multiple, discrete, autonomous locations, such that datagrams sent are routed to one of several available locations".

Dyncast: Dynamic Anycast, taking the dynamic nature of computing resource metrics into account to steer an anycast-like decision in sending an incoming service request.

3. Desirable System Characteristics and Requirements

In the following, we outline the desirable characteristics of a system to overcome the observed problems in [I-D.liu-dyncast-ps-usecases] for the realization of the use cases described in that document.

3.1. 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 can potentially 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.Since a client will use the service identifier as the destination addressing, mapping of the service identifier to the unicast address will need to happen in-band, considering the metrics for selection to make this selection service-specific. From an addressing perspective, a desirable system for the realization of the use cases described in that document.

o MUST provide a discovery and mapping methodology for the in-band mapping of the service identifier (an anycast address) to a specific unicast address.

3.2. Instance Affinity

A routing relation between a client and a service exists not at the packet but at the service request level in the sense that one or more service requests, possibly consisting of one or many more routing-level packets, must be ensured to be sent to said service.Each service may be provided by one or more service instances, each providing equivalent service functionality to their respective clients, while those service instances may be deployed at different locations in the network. With that, the routing problem becomes one between the client and a selected service instance for at least the duration of the service-level request, but possibly more than just one request.

This relationship between the client and the chosen service instance is described as "instance affinity" in the following, where the "affinity" spans across the aforementioned one or more service requests. This impacts the routing decision to be taken in that the normal packet level communication, i.e., each packet is forwarded individually based on the forwarding table at the time, will need extending with the notion of instance affinity since otherwise individual packets may be sent to different places when the network status changes, possibly segmenting individual requests and breaking service-level semantics.

The nature of this affinity is highly dependent on the nature of the specific service. The minimal affinity of a single request represents a stateless service, where each service request may be responded to without any state being held at the service instance for fulfilling the request. Providing any necessary information/state in-band as part of the service request, e.g., in the form of a multi-form body in an HTTP request or through the URL provided as part of the request, is one way to achieve such stateless nature. Alternatively, the affinity to a particular service instance may span more than one request, as in our VR example in [I-D.liu-dyncast-ps-usecases], where previous client input is needed to render subsequent frames. Therefore, a desirable system

o MUST maintain "instance affinity" which MAY span one or more service requests, i.e., all the packets from the same flow MUST go to the same service instance.

3.3. Proper Runtime-state Granularity and Keeping

The instance affinity, as outlined in Section 3.2, requires a client and the chosen service instance to keep persistent relationship across one or more service requests. For a multi-request session, this determines that the mapping logic has to consistently pick up the same service instance. This type of affinity can be normally achieved by deploying a mapping device to keep in-place all the necessary states. However, a client, e.g., a mobile UE, has generally many applications running. If all, or majority, of the applications request the dyncast-like services, then the runtime states that need to be created and accordingly maintained would require high granularity. In the extreme scenario, this granular requirement could reach the level of per-UE per-APP per-(sub)flow with regard to a service instance.

Evidently, these fine-granular runtime states can potentially become heavy burden for network devices if they have to dynamically create and maintain them. On the other hand, it is not appropriate either to place the state-keeping task on clients themselves. Therefore, a desirable system

o MUST avoid keeping fine runtime-state granularity in network nodes in order to achieve instance affinity.

o MUST provide mechanism to free clients from maintaining granular runtime-states in order to achieve instance affinity.

3.4. Encoding Metrics

As outlined in the scenarios in [I-D.liu-dyncast-ps-usecases], metrics can have many different semantics, particularly if considered to be service- specific. Even the notion of a "computing load" metric may be computed in many different ways. What is crucial, however, is the representation and encoding of that metric when being conveyed to the routing fabric in order for the routing elements to act upon those metrics. Such representation may entail information on the semantics of the metric or it may be purely one or more semantic-free numerals. Agreement of the chosen representation among all service and network elements participating in the service-specific routing decision is important. Specifically, a desirable system

o MUST agree on the service-specific metrics and their representation between service elements in the participating edges in the network and network elements acting upon them.

o MAY obfuscate the specific semantic of the metric to preserve privacy of the service provider information towards the network provider.

o MAY include routing protocol metrics

3.5. Signaling Metrics

The aforementioned representation of metrics needs conveyance to the network elements that will need to act upon them. Depending on the service-specific decision logic, one or more metrics will need to be conveyed. Problems to be addressed here may be that of loop avoidance of any advertisement of metrics as well as the frequency of such conveyance and therefore the overall load that the signaling may add to the overall network traffic. While existing routing protocols may serve as a baseline for signaling metrics, other means to convey the metrics can equally be realized. Specifically, a desirable system

o MUST provide mechanisms to signal the metrics for using in routing decisions

o MUST realize means for rate control for signaling of metrics

o MUST implement mechanisms for loop avoidance in signaling metrics, when necessary

3.6. Using Metrics in Routing Decisions

Metrics being conveyed, as outlined in Section 3.4, in the agreed manner, as outlined in Section 3.3, will ultimately need suitable action in the routers of the network. Routing decisions can be manifold, possibly including (i) min or max over all metrics, (ii) extending previous action with a random or first choice when more than one min/max entry found, (iii) weighted round robin of all entries, among others. It is important for the proper work of the service-specific routing decision, that it is understood to both network and service provider, which action (out of a possible set of supported actions) is to be used for a particular set of metrics. Specifically, a desirable system

Further, different network nodes, e.g., routers, switches, etc., bear diversified capabilities even in the same routing domain, let alone in different administrative domains. So, the service-specific metrics that have been adopted by some nodes might not be supported by others, either due to technical reasons, administrative reasons, or something else. There could be some scenario that a node supporting service-specific metrics might prefer some type of metrics

to others [3GPP-TR22.847], or, in another scenario, even not utilize any at all. Therefore, there must exist flexibility in term of metrics handling and routing decisions in a network.

o MUST specify a default action to be taken, if more than one action possible

o MUST allow a network node not supporting service-specific metrics to interoperate with the supporting ones, i.e., providing backward compatibility.

o SHOULD allow the prioritization of using the service-specific metrics when compared to the currently widely-used networking metrics, like bandwidth, delay, loss, etc.

o SHOULD enable other alternative actions to be taken. (1)Any solution MUST provide appropriate signaling of the desired action to the router. For this, the action MAY be signaled in combination with signaling the metric (see Section 3.4). (2)Any solution SHOULD allow associating the desired action to a specific service identifier.

3.7. Supporting Service Dynamism

Network cost in the current routing system usually does not change very frequently. However, computing load and service-specific metrics in general can be highly dynamic, e.g., changing rapidly with the number of sessions, CPU/GPU utilization and memory space. It has to be determined at what interval or events such information needs to be distributed among edges. More frequent distribution of more accurate synchronization may result in more overhead in terms of signaling.

Choosing the least path cost is the most common rule in routing. However, the logic does not work well when routing should be aware of service-specific metrics. Choosing the least computing load may result in oscillation. The least loaded edge can quickly be flooded by the huge number of new computing demands and soon become overloaded with tidal effects possibly following.

Generally, a single instance may have very dynamic resource availability over time in order to serve service requests. This availability may be affected by computing resource capability and load, network path quality, and others. The balancing mechanisms should adapt to the service dynamism quickly and seamlessly. With this, the relationship between a single client and the set of possible service instances may possibly be very dynamic in that one request that is being dispatched to instance A may be followed by a request that is being dispatched to instance B and so on, generally within the notion of the service-specific service affinity discussed before in Section 3.2. With this in mind, a desirable system

o MUST support the dynamics of metrics changing on, e.g., a per flow basis, without violating the metrics defined in the selection of the specific service instance, while taking into account the requirements for the signaling of metrics and routing decision (see Section 3.4 and 3.5).

4. Conclusion

This document presents high-level requirements for solutions to Dyncast, where the architecture should address how to distribute the resource information and how to assure instance affinity in an anycast based service addressing environment, while realizing appropriate routing actions to satisfy the metrics provided.

5. Security Considerations

TBD

6. IANA Considerations

No IANA action is required so far.

7. Contributors

The following people have substantially contributed to this document:

Peter Willis
BT

8. Informative References

[RFC4786] Abley, J. and K. Lindqvist, "Operation of Anycast Services", BCP 126, RFC 4786, DOI 10.17487/RFC4786, December 2006, <https://www.rfc-editor.org/info/rfc4786>.

[I-D.liu-dyncast-ps-usecases]
Liu, P., Willis, P., Trossen, D., and C. Li, "Dynamic-Anycast (Dyncast) Use Cases & Problem Statement", Work in Progress, Internet-Draft, draft-liu-dyncast-ps-usecases-02, 17 January 2022, <https://www.ietf.org/archive/id/draft-liu-dyncast-ps-usecases-02.txt>.

[TR22.874] 3GPP, "Study on traffic characteristics and performance requirements for AI/ML model transfer in 5GS (Release 18)", 2020.

Acknowledgements

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

Authors' Addresses

Peng Liu
China Mobile
Email: liupengyjy@chinamobile.com

Tianji Jiang
China Mobile
Email: jiangtianji@chinamobile.com

Philip Eardley
British Telecom
Email: philip.eardley@bt.com

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

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