

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

L. Geng
P. Liu
China Mobile
P. Willis
BT
October 30, 2020

Dynamic-Anycast in Compute First Networking (CFN-Dyncast) Use Cases and Problem Statement

[draft-geng-rtgwg-cfn-dyncast-ps-usecase-00](#)

Abstract

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

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

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

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

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

Internet-Draft

CFN-Dyncast Use Cases and Problems

October 2020

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

This Internet-Draft will expire on May 3, 2021.

Copyright Notice

Copyright (c) 2020 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Introduction	2
2.	Definition of Terms	4
3.	Main Use-Cases	4
3.1.	Cloud Based Recognition in Augmented Reality (AR)	4
3.2.	Connected Car	5
3.3.	Cloud Virtual Reality (VR)	5
4.	Requirements	6
5.	Problems Statement	6
5.1.	Anycast based service addressing methodology	7
5.2.	Flow affinity	7
5.3.	Computing Aware Routing	8
6.	Summary	8
7.	Security Considerations	9
8.	IANA Considerations	9
9.	Informative References	9
	Acknowledgements	9
	Authors' Addresses	9

[1.](#) Introduction

Edge computing aims to provide better response times and transfer

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

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

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

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

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

There are some dynamic ways which tries to distribute the request to the server with the best available resources and minimal load. They usually require L4-L7 handling of the packet processing. It is not

an efficient approach for huge number of short connections. At the same time, such approaches can hardly get network status in real time. Therefore the choice of the service node is almost entirely determined by the computing status, rather than the comprehensive consideration of both computing and network.

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

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

[2.](#) Definition of Terms

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

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

[3.](#) Main Use-Cases

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

[3.1.](#) Cloud Based Recognition in Augmented Reality (AR)

In AR environment, the end device captures the images via cameras and sends out the computing intensive service demand. Normally service nodes at the edge are responsible for tasks with medium computational

complexity or low latency requirement like object detection, feature extraction and template matching, and service nodes at cloud are responsible for the most intensive computational tasks like object recognition or latency non-sensitive tasks like AI based model training. The end device hence only handles the tasks like target tracking and image display, thereby reducing the computing load of the client.

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

[3.2.](#) Connected Car

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

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

[3.3.](#) Cloud Virtual Reality (VR)

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

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

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

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

[4.](#) Requirements

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

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

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

5. Problems Statement

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

- o Use the least network cost as metric to select the edge. Issue: Computing information is a key to be considered in edge computing, and it is not included here.
- o Use geographical location deduced from IP prefix, pick the closest edge. Issue: Edges are not so far apart in edge computing scenario. Either hard to be deduced from IP address or the location is not the key distinguisher.
- o Health check in an infrequent base (>1s) to reflect the service node status, and switch when fail-over. Issue: Health check is very different from computing status information of service instance. It is too coarse granularity.
- o Application layer randomly picks or uses round-robin way to pick a service node. Issue: It may share the load across multiple service instances in terms of the computing capacity, the network cost variance is barely considered. Edges can be deployed in

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

- o Global resolver and early binding (DNS-based load balancing): Client queries a global resolver or load balancer first and gets the exact server's address. And then steer traffic using that address as binding address. It is called early binding because an explicit binding address query has to be performed before sending

user data. Issue: Firstly, it clashes with the service dynamism. Current resolver does not have the capability of such high frequent change of indirection to new instance based on the frequent change of each service instance. Secondly, edge computing flow can be short. One or two round trip would be completed. Out-of-band query for specific server address has high overhead as it takes one more round trips. As discussed in section 5.4 of [[I-D.sarathchandra-coin-appcentres](#)], the flexible re-routing to appropriate service instances out of a pool of available ones faces significant challenges when utilizing DNS for this purpose.

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

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

[5.1.](#) Anycast based service addressing methodology

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

[5.2.](#) Flow affinity

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

data plane must maintain flow affinity. All the packets from the

same flow should go to the same service node.

5.3. Computing Aware Routing

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

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

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

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

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

6. Summary

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

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

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

7. Security Considerations

TBD

8. IANA Considerations

No IANA action is required so far.

9. Informative References

[I-D.sarathchandra-coin-appcentres]

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

Acknowledgements

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

Authors' Addresses

Liang Geng
China Mobile

Email: gengliang@chinamobile.com

Peng Liu
China Mobile

Email: liupengyjy@chinamobile.com

Peter Willis
BT

Email: peter.j.willis@bt.com

