Workgroup: rtgwg
Internet-Draft:
draft-liu-dyncast-ps-usecases-04
Published: 8 July 2022
Intended Status: Informational
Expires: 9 January 2023
Authors: P. Liu
China Mobile
P. Eardley
BT
D. Trossen
Huawei Technologies
M. Boucadair
Orange
LM. Contreras
Telefonica
C. Li
Huawei Technologies
Y. Li
Huawei Technologies

# Dynamic-Anycast (Dyncast) Problem Statement and Use Cases

## 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

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

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute

working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

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

This Internet-Draft will expire on 9 January 2023.



**Table of Contents**



## 1. Introduction

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.

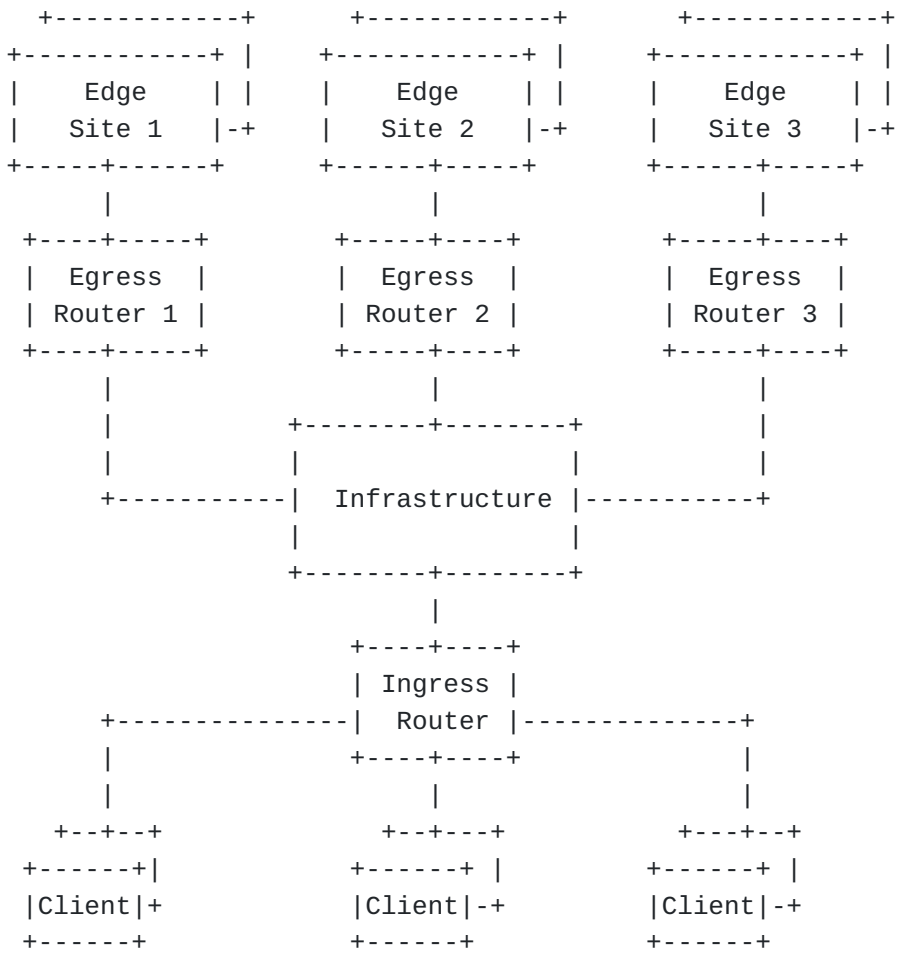


Figure 1: CAN Use Case Model

### 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.

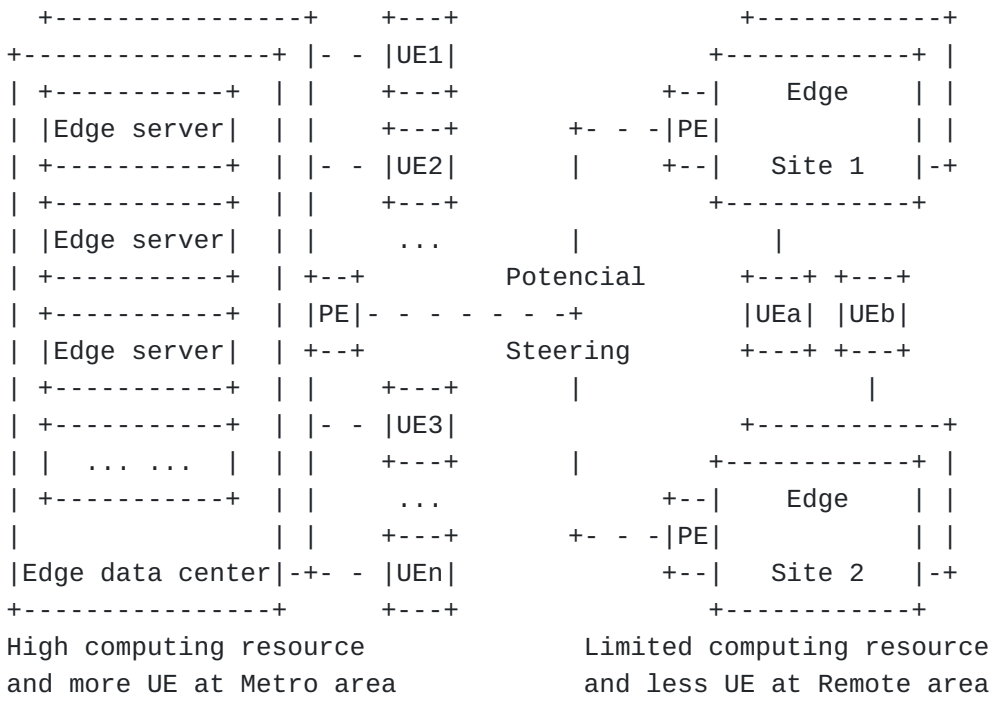


Figure 2: Common Deployment of Edge Sites

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.

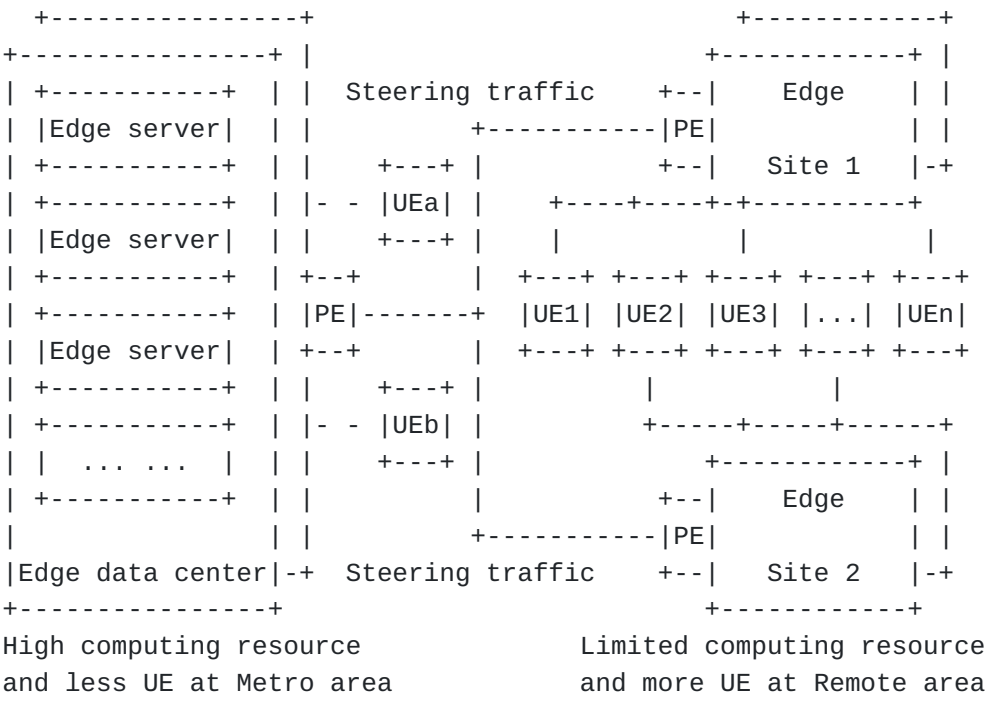


Figure 3: Steering Traffic among Edge Sites

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.

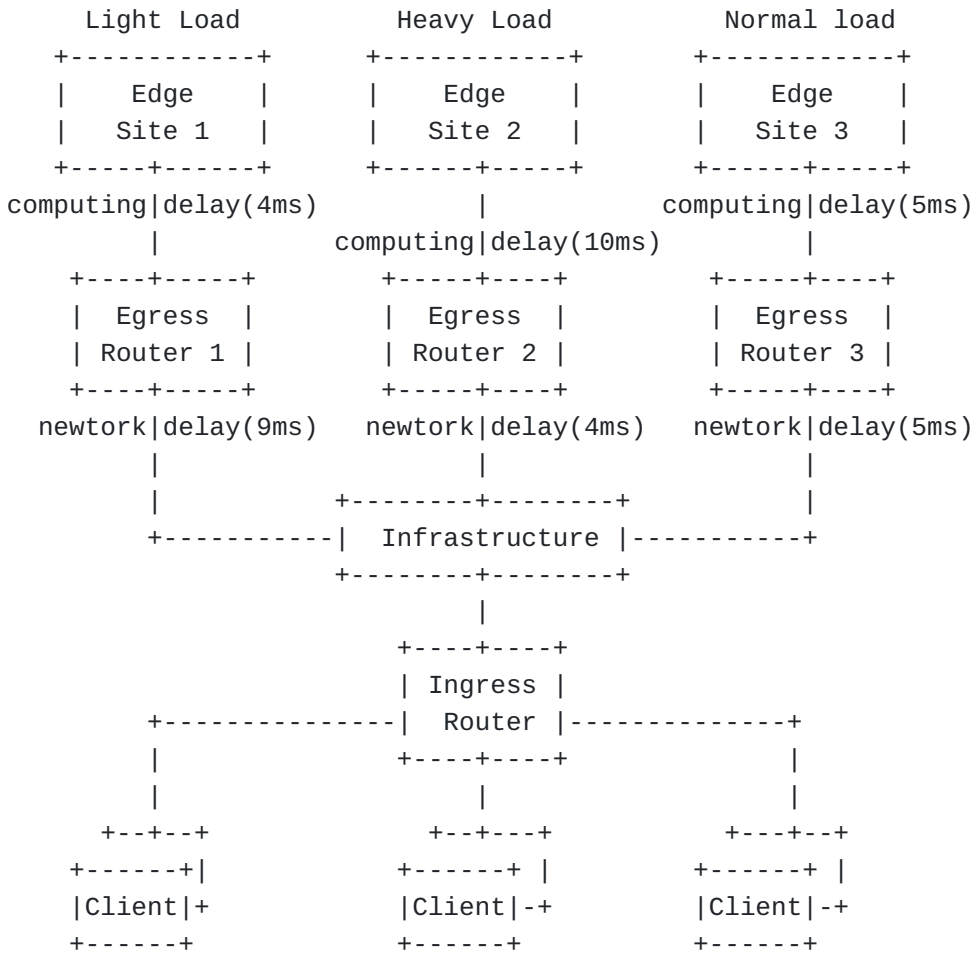


Figure 4: Computing-Aware AR or VR

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