

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

P. Liu
China Mobile
P. Eardley
British Telecom
D. Trossen
Huawei Technologies
M. Boucadair
Orange
LM. Contreras
Telefonica
C. Li
Huawei Technologies
7 March 2022

Dynamic-Anycast (Dyncast) Use Cases and Problem Statement
draft-liu-dyncast-ps-usecases-03

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.

This document provides an overview of scenarios and problems associated with realizing such scenarios, identifying key engineering investigation areas which require adequate architectures and protocols to achieve balanced computing and networking resource utilization among facilities providing the services.

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 8 September 2022.

Copyright Notice

Copyright (c) 2022 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 Revised BSD License text as described in Section 4.e of the [Trust Legal Provisions](#) and are provided without warranty as described in the Revised BSD License.

Table of Contents

1.	Introduction	3
2.	Definition of Terms	4
3.	Sample Use Cases	5
3.1.	Cloud Virtual Reality (VR) or Augmented Reality (AR) . .	6
3.2.	Intelligent Transportation	8
3.3.	Digital Twin	9
4.	Problems in Existing Solutions	10
4.1.	Dynamicity of Relations	10
4.2.	Efficiency	12
4.3.	Complexity and Accuracy	12
4.4.	Metric Exposure and Use	13
4.5.	Security	13
4.6.	Changes to Infrastructure	14
5.	Conclusion	14
6.	Security Considerations	15

7.	IANA Considerations	15
8.	Contributors	15
9.	Informative References	15
	Acknowledgements	16
	Authors' Addresses	16

[1.](#) Introduction

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 is an emerging requirement that multiple edge sites (called "edges", for for short) are deployed at different locations to provide a 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 behaving as service nodes. 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.

Some user devices are battery-dependent. Offloading computation intensive processing to the edge can save battery power. Moreover, the edge may use a 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.

At the same time, with new technologies such as serverless computing and container based virtual functions, the service node at an edge can be easily created and terminated in a sub-second scale, which in turn changes the availability of a computing resources for a service dramatically over time, therefore impacting the possibly "best" decision on where to send a service request from a client.

Traditional techniques to manage the overall load balancing process of clients issuing requests include choose-the-closest or round-robin. Those solutions are relatively static, which may cause an

unbalanced distribution in terms of network load and computational load among available sources. For example, DNS-based load balancing usually configures a domain in the Domain Name System (DNS) such that client requests to that domain name are distributed across a group of servers. It usually provides several IP addresses for a domain name.

There are some dynamic solutions to distribute the requests to the server that best fits a service-specific metric, such as the best available resources and minimal load. They usually require Layer 4 - Layer 7 handling of the packet processing, such as through DNS-based or indirection servers. Such an approach is inefficient for large number of short connections. At the same time, such approaches can

often not retrieve the desired metric, such as the network status, in real time. Therefore, the choice of the service node is almost entirely determined by the computing status, rather than the comprehensive considerations of both computing and network metrics or makes rather long-term decisions due to the (upper layer) overhead in the decision making itself.

Distributing service requests to a specific service having multiple instances attached to multiple edges, while taking into account computing as well as service-specific metrics in the distribution decision, is seen as a dynamic anycast (or "dyncast", for short) problem of sending service requests, without prescribing the use of a routing solution.

As a problem statement, 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. Those key areas target the identification of candidate solution components.

[2.](#) Definition of Terms

This document makes use of the following 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 behavior, no matter where those service instances are running.

Anycast: An addressing and packet forwarding approach that assigns an "anycast" identifier for one or more service instances to which requests to an "anycast" identifier could be routed/forwarded, 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.](#) Sample 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 requirements and ensure better user experience.

Before outlining the use cases, however, let us describe a basic model that we assume through which those use cases are being realized. This model justifies the choice of the terminology introduced in [Section 2](#).

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

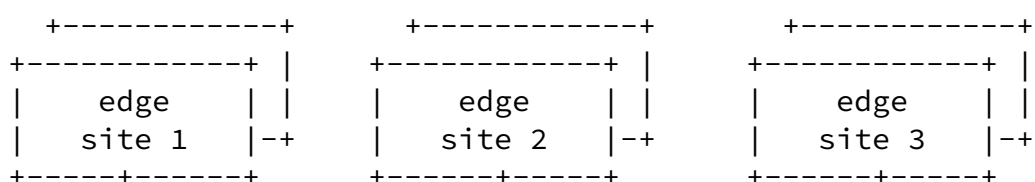
available service instances that realize the requested service. Each service instance may act as a client towards another service, thereby seeing its own outbound traffic steered to a suitable service instance of the request service and so on, achieving service composition and chaining as a result.

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.

Apart from the input into the traffic steering decision, under the aforementioned constraint of possible client mobility, its realization may differ in terms of the layer of the protocol stack at which the needed operations for the decision are implemented. Possible layers are application, transport, or network layers. [Section 4](#) discusses some choice realization issues.

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



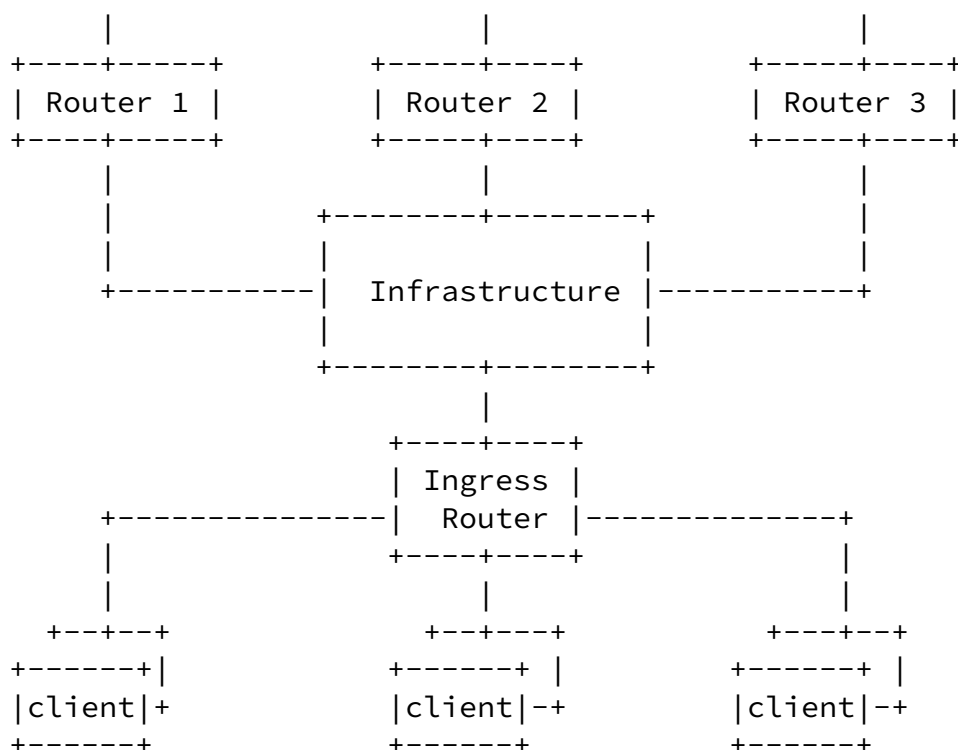


Figure 1: Dyncast Use Case Model

3.1. Cloud Virtual Reality (VR) or Augmented Reality (AR)

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, (ii) image/frame rendering delay, (iii) display refresh delay, and (iv) network delay. With upcoming high display refresh rate (e.g., 144Hz) and GPU resources being used for frame rendering, we can expect an upper bound of roughly 5ms for the round-trip latency in these scenarios, which is close to the frame rendering computing delay.

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.

3.2. 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.

3.3. 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 losing 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.

Digital twin for networks[I-D.zhou-nmrg-digitaltwin-network-concepts] has also been proposed recently. It is to introduce digital twin technology into the network to build a network system with physical network entities and virtual twins, which can be mapped in real time. The goal of digital twin network will be applied not only to industrial Internet, but also to operator network. When the network is large, it needs real-time scheduling ability, more efficient and accurate data collection and modeling, and promote the automation, intelligent operation and maintenance and upgrading of the network.

[4.](#) Problems in Existing Solutions

There are a number of problems that may occur when realizing the use cases listed in the previous section. This section suggests a classification for those problems to aid the possible identification of solution components for addressing them.

[4.1.](#) Dynamicity of Relations

The mapping from a service identifier to a specific service instance that may execute the service for a client usually happens through resolving the service identification into a specific IP address at which the service instance is reachable.

Application layer solutions can be foreseen, using an application server to resolve binding updates. While the viability of these solutions will generally depend on the additional latency that is being introduced by the resolution via said application server, frequencies down to changing relations every few (or indeed EVERY) service requests is seen as difficult to be viable.

Message brokers, however, could be used, dispatching incoming service requests from clients to a suitable service instance, where such dispatching could be controlled by service-specific metrics, such as computing load. The introduction of such brokers, however, may lead to adverse effects on efficiency, specifically when it comes to additional latencies due to the necessary communication with the broker; we discuss this problem separately in the next subsection.

DNS[RFC1035] realizes an 'early binding' to explicitly bind from the service identification to the network address before sending user data, so the client creates an 'instance affinity' for the service identifier that binds the client to the resolved service instance address, which could also realize the load balancing.

However, we can foresee scenarios in which such 'instance affinity' may change very frequently, possibly even at the level of each service request. One such driver may be frequently changing metrics for the decision making, such as latency and load of the involved service instance. Also client mobility creates a natural/physical dynamicity with the result that 'better' service instances may become available and, vice versa, previous assignments of the client to a service instance may be less optimal, leading to reduced performance, such as through increased latency.

DNS is not designed for this level of dynamicity. Updates to the

mapping between service identifier to service instance address cannot be pushed quickly enough into the DNS that takes several minutes updates to propagate, and clients would need to frequently resolve the original binding. If try to DNS to meet this level of dynamicity, frequent resolving of the same service name would likely lead to an overload of the it. These issues are also discussed in Section 5.4 of [[I-D.sarathchandra-coin-appcentres](#)].

A solution that leaves the dispatching of service requests entirely to the client may be possible to achieve the needed dynamicity, but with the drawback that the individual destinations, i.e., the network identifiers for each service instance, must be known to the client for doing so. While this may be viable for certain applications, it cannot generally scale with a large number of clients. Furthermore, it may be undesirable for every client to know all available service instance identifiers, e.g., for reasons of not wanting to expose this

information to clients from the perspective of the service provider but also, again, for scalability reasons if the number of service instances is very high.

Existing solutions exhibit limitations in providing dynamic 'instance affinity', those limitations being inherently linked to the design used for the mapping between the service identifier and the address of the service instance, particularly when relying on an indirection point in the form of a resolution or load balancing server. These limitations may lead to 'instance affinity' to last many requests or even for the entire session between the client and the service, which may be undesirable from the service provider perspective in terms of best balance requests across many service instances.

[4.2.](#) Efficiency

The use of external resolvers, such as application layer repositories in general, also affects the efficiency of the overall service request. Additional signaling is required between client and resolver, either through the application layer solution, which not only leads to more messaging but also to increased latency for the additional resolution. Accommodating smaller instance affinities increases this additional signaling but also the latencies experienced, overall impacting the efficiency of the overall service transaction.

As mentioned in the previous subsection, broker systems could be used to allow for dispatching service requests to different service instances at high dynamicity. However, the usage of such broker inevitably introduces 'path stretch' compared to the possible direct path between client and service instance, increasing the overall flow completion time.

Existing solutions may introduce additional latencies and inefficiencies in packet transmission due to the need for additional resolution steps or indirection points; and will lead to the accuracy problems to select the appropriate edge.

[4.3.](#) Complexity and Accuracy

As we can see from the discussion on efficiency in the previous subsection, the time when external resolvers collect the necessary information and deal with it to select the edge nodes, the network and computing resource status may change already. So any additional control decision on which service instance to choose for which incoming service request requires careful planning to keep potential inefficiencies, caused by additional latencies and path stretch, at a minimum. Additional control plane elements, such as brokers, are

usually neither well nor optimally placed in relation to the data path that the service request will ultimately traverse.

Existing solutions require careful planning for the placement of necessary control plane functions in relation to the resulting data plane traffic to improve the accuracy; a problem often intractable in scenarios of varying service demand.

[4.4.](#) Metric Exposure and Use

Some systems may use the geographical location, as deduced from IP prefix, to pick the closest edge. The issue here may be that edges may not be far apart in edge computing deployments, while it may also be hard to deduce geo-location from IP addresses. Furthermore, the geo-location may not be the key distinguishing metric to be considered, particularly if geographic co-location does not necessarily mean network topology co-location. Also, "closer geographically" does not consider the computing load of possible

closer yet more loaded nodes, consequently leading to possibly worse performance for the end user.

Solutions may also perform 'health checks' on an infrequent base (>1s) to reflect the service node status and switch in fail-over situations. Health checks, however, inadequately reflect an overall computing status of a service instance. It may therefore not reflect at all the decision basis a suitable service instance, e.g., based on the number of ongoing sessions as an indicator of load. Infrequent checks may also be too coarse in granularity, e.g., for supporting mobility-induced dynamics such as the connected car scenario of [Section 3.2](#).

Service brokers may use richer computing metrics (such as load) but may lack the necessary network metrics.

Existing solutions lack the necessary information to make the right decision on the selection of the suitable service instance due to the limited semantic or due to information not being exposed across boundaries between, e.g., service and network provider.

[4.5](#). Security

Resolution systems opens up two vectors of attack, namely attacking the mapping system itself, as well as attacking the service instance directly after having been resolved. The latter is particularly an issue for a service provider who may deploy significant service infrastructure since the resolved IP addresses will enable the client to directly attack the service instance but also infer (over time) information about available service instances in the service

infrastructure with the possibility of even wider and coordinated Denial-of-Service (DoS) attacks.

Broker systems may prevent this ability by relying on a pure service identifier only for the client to broker communication, thereby hiding the direct communication to the service instance albeit at the expense of the additional latency and inefficiencies discussed in [Section 4.1](#) and 4.2. DoS attacks here would be entirely limited to the broker system only since the service instance is hidden by the broker.

Existing solutions may expose control as well as data plane to the possibility of a distributed Denial-of-Service attack on the resolution system as well as service instance. Localizing the attack to the data plane ingress point would be desirable from the perspective of securing service request routing, which is not achieved by existing solutions.

[4.6.](#) Changes to Infrastructure

Dedicated resolution systems, such as the DNS or broker-based systems, require appropriate investments into their deployment. While the DNS is an inherent part of the Internet infrastructure, its inability to deal with the dynamicity in service instance relations, as discussed in [Section 4.1](#), may either require significant changes to the DNS or the establishment of a separate infrastructure to support the needed dynamicity. In a manner, the efforts on Multi-Access Edge Computing [[MEC](#)], are proposing such additional infrastructure albeit not solely for solving the problem of suitably dispatching service requests to service instances (or application servers, as called in [[MEC](#)]).

Existing solutions may expose control as well as data plane to the possibility of a distributed Denial-of-Service attack on the resolution system as well as service instance. Localizing the attack to the data plane ingress point would be desirable from the perspective of securing service request routing, which is not achieved by existing solutions.

[5.](#) Conclusion

This document presents use cases 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, 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, with a number of observed problems in existing

solutions. The use cases as well as the categorization of the observed problems may start the process of determining how they are best satisfied within the IETF protocol suite or through suitable extensions to that protocol suite.

6. Security Considerations

Section 4.5 discusses some security considerations.

7. IANA Considerations

This document does not make any IANA request.

8. Contributors

The following people have substantially contributed to this document:

Peter Willis
BT

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>>.
- [RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, [RFC 1035](#), DOI 10.17487/RFC1035, November 1987, <<https://www.rfc-editor.org/info/rfc1035>>.
- [I-D.zhou-nmrg-digitaltwin-network-concepts]
Zhou, C., Yang, H., Duan, X., Lopez, D., Pastor, A., Wu, Q., Boucadair, M., and C. Jacquenet, "Digital Twin Network: Concepts and Reference Architecture", Work in Progress, Internet-Draft, [draft-zhou-nmrg-digitaltwin-network-concepts-07](#), 5 March 2022, <<https://www.ietf.org/archive/id/draft-zhou-nmrg-digitaltwin-network-concepts-07.txt>>.

[I-D.sarathchandra-coin-appcentres]

Trossen, D., Sarathchandra, C., and M. Boniface, "In-Network Computing for App-Centric Micro-Services", Work in Progress, Internet-Draft, [draft-sarathchandra-coin-appcentres-04](https://www.ietf.org/archive/id/draft-sarathchandra-coin-appcentres-04), 26 January 2021, <<https://www.ietf.org/archive/id/draft-sarathchandra-coin-appcentres-04.txt>>.

[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 Yizhou Li, 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
British Telecom
Email: philip.eardley@bt.com

Dirk Trossen

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

