Workgroup: cats
Internet-Draft:
draft-ietf-cats-usecases-requirements-00
Published: 24 July 2023
Intended Status: Informational
Expires: 25 January 2024
Authors: K. Yao
China Mobile
D. Trossen
Huawei Technologies
M. Boucadair
Orange
LM. Contreras
Telefonica
H. Shi
Huawei Technologies
Y. Li
Huawei Technologies
S. Zhang
China Unicom

Computing-Aware Traffic Steering (CATS) Problem Statement, Use Cases, and Requirements

Abstract

Distributed computing is a tool that service providers can use to achieve better service response time and optimized energy consumption. In such a distributed computing environment, providing services by utilizing computing resources hosted in various computing facilities aids support of services such as computationally intensive and delay sensitive services. Ideally, compute services are balanced across servers and network resources to enable higher throughput and lower response times. To achieve this, the choice of server and network resources should consider metrics that are oriented towards compute capabilities and resources instead of simply dispatching the service requests in a static way or optimizing solely on connectivity metrics. The process of selecting servers or service instance locations, and of directing traffic to them on chosen network resources is called "Computing-Aware Traffic Steering" (CATS).

This document provides the problem statement and the typical scenarios for CATS, which shows the necessity of considering more factors when steering traffic to the appropriate computing resource to best meet the customer's expectations and deliver the requested service.

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 25 January 2024.

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. Network operators have extended their capabilities by complementing their network infrastructure by developing CDN capabilities, particularly in edge sites. In addition, more computing resource are deployed at these edge sites as well.

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, Augmented Reality(AR)/Virtual Reality(VR), live broadcast and so on. On the other hand, users want 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. There are millions of home gateways, thousands of base stations, and hundreds of central offices in a city that could serve as compute-capable nodes to deliver a service. Note that not all of these nodes would be considered as edge nodes in some views of the network, but they can all provide computing resources to enable a service.

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

Depending on the locations of computing resource and their capacity, different amounts of resource from different locations can be used to deliver a service. At peak hours, computing resources closest to a client might not be sufficient to handle all the incoming service requests. Longer response times or even dropping of requests could be experienced by users. Increasing the computing resources hosted at each location to the potential maximum capacity is neither feasible nor economically viable in many cases. Offloading computation intensive processing to the user devices would place huge pressure on local resources such as the 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.

Service providers often have their own server sites, many of which have been enhanced to support computing services. A service instance deployed at a single site might not provide sufficient capacity to fully guarantee the quality of service required by a customer. Instead, the same service can be deployed at multiple sites for better availability and scalability. Furthermore, it is desirable to balance the load across all service instances to improve throughput. For this, traffic needs to be steered to the 'best' service instance according to information that may include current computing load, where the notion of 'best' may highly depend on the application demands.

A particular example is the popular and pervasive 5G Mobile Edge Computing(MEC) service. In 5G MEC, the Uplink Classifier(UL-CL) functionality of User Plane Functions(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 Data Network(DN), operators' DN, or even 3rd-party's DN. Through possibly using some 'intelligent' criteria, this could warrant the selection of resources with either low, high-computational power or all-involved requirements.

This document describes sample usage scenarios that drive CATS requirements and will help to identify candidate solution architectures and solutions.

2. Definition of Terms

This document makes use of the following terms:

Service: An offering provided by a service provider, similar to the notion of a 'service function' in [RFC7665], which may or may not be of composite nature but appears in the problem space of CATS as a single service to which traffic needs to be steered.

Service instance: A run-time environment (e.g., a server or a process on a server) that makes a service available. A particular service could be made available at multiple service instances at the same or different 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.

Even though this document is not a protocol specification, it makes use of upper case key words to define requirements unambiguously. 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. 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 site with the same application in the city/province/state, a number of representative cities have deployed multi-edge sites and the typical applications, and there are more edge sites to be deployed in the future. Before deploying edge sites, there are some factors that need to be considered, such as:

o The exsiting infrastructure capacities, which could be updated to edge sites, e.g. operators' machine room.

o The amount and frequency of computing resource that is needed.

o The network resource status linked to computing resource.

To improve the effectiveness of service deployment, the problem of how to choose the optimal edge node on which to deploy services needs to be solved. [I-D.contreras-alto-service-edge] introduces considerations for how to deploy applications or functions to the edge, such as the type of instance, optional storage extension, optional hardware acceleration characteristics, and the compute flavor of CPU/GPU, etc. More network and service factors may also be considered, such as:

o Network and computing resource topology: The overall consideration of network access, connectivity, path protection or redundancy, and the location and overall distribution of computing resources in the network, and the relative position within the network topology.

o Location: The number of users, the differentiation of service types, and the number of connections requested by users, etc. For edge nodes located in populous area with a large number of users and service requests, service duplication could be deployed more than in other areas.

o Capacity of multiple edge nodes: Not only the capacity of a single node, but also the total number of requests that can be processed by the resource pool composed of multiple nodes.

o Service category: For example, whether the service is a multi-user interaction, such as video conferencing, or games, or whether it just resource acquisition, such as video viewing. ALTO [RFC7285] can help to obtain one or more of the above pieces of information, so as

to provide suggestions or formulate principles and strategies for service deployment.

This information could be collected periodically, and could record the total consumption of computing resources, or the total number of sessions accessed. This would indicate whether additional service instances need to be deployed. Unlike the scheduling of service requests, service deployment should follow the principle of proximity to place new service instances near to customer sites that will request them. If the resources are insufficient to support new instances, the operator can be informed to increase the hardware resources.

In general, the choice of where to locate service instances and when to create new ones in order to provide the right levels of resource to support user demands is important in building a network that supports computing services. However, those aspects are out of scope for CATS and are left for consideration in another document.

3.2. Traffic Steering among Edges Sites and Service Instances

This section describes how existing edge computing systems do not provide all of the support needed for real-time or near-real-time services, and how it is necessary to steer traffic to different sites considering mobility of people, different time slots, events, server loads, and network capabilities, etc.

In edge computing, the computing resources and network resources are considered when deploying edge sites and services. Traffic is steered to an edge site that is "closest" or to one of a few "close" sites using load-balancing. But the "closest" site is not always the "best" as the status of computing resources and of the network may vary as follows:

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.

o The network path to the closest site might not provide the necessary network characteristics, such as low latency or high throughput.

To address these issues some enhancements are needed to steer traffic to sites that can support the requested services.

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 edge router.

When a client issues a service request for a required service, the request is steered to one of the available service instances. 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, latency experienced or similar, for a more dynamic selection of a suitable service instance.

It is important to note that clients may move. This means that the service instance that was "best" at one moment might no longer be best when a new service request is issued. This creates a (physical) dynamicity that will need to be catered for in addition to the changes in server and network load.

Figure 1 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 User Equipments(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.

Applications to meet service demands 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 be needed just for special service request or some small scheduling demand.

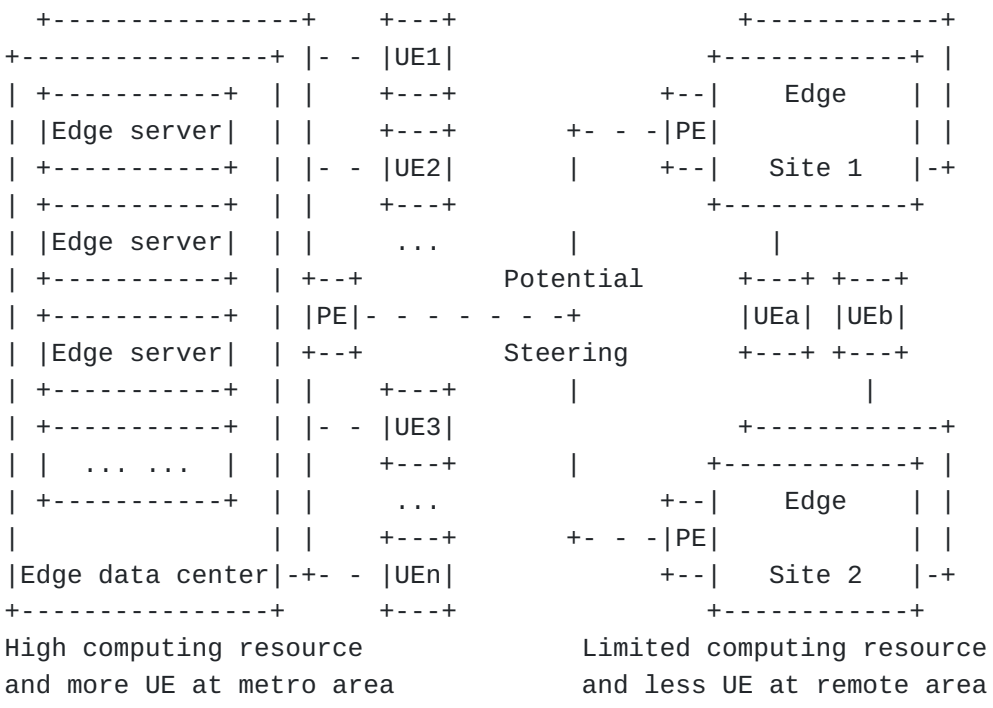

Figure 1: Common Deployment of Edge Sites

Figure 2 shows that during non-working hours, 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 many people to request services at the remote area, but with the limited computing resource, moreover, as the people move from the metro area to the remote area, the edge sites that serve common services will also change, so it may be necessary to steer some traffic back to the metro data center.

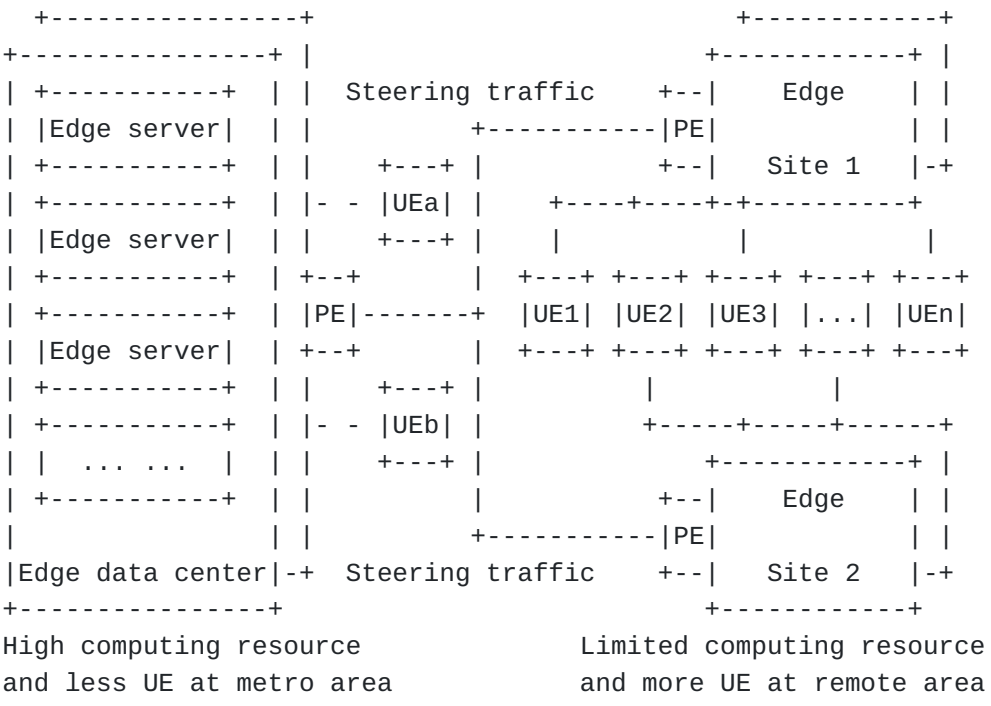

Figure 2: Steering Traffic among Edge Sites

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 dynamicity.

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 use cases in the following section require both low latency and high computing resource usage or specific computing hardware capabilities (such as local GPU); hence joint optimization of network and computing resource is needed to guarantee the Quality of Experience(QoE).

4. Use Cases

This section presents a non-exhaustive set of use cases which would benefit from the dynamic selection of service instances and the steering of traffic to those service instances.

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) round trip 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 3 shows, the client could request any service instance among 3 edge sites. The delay of client could be same, and the differences of different 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.

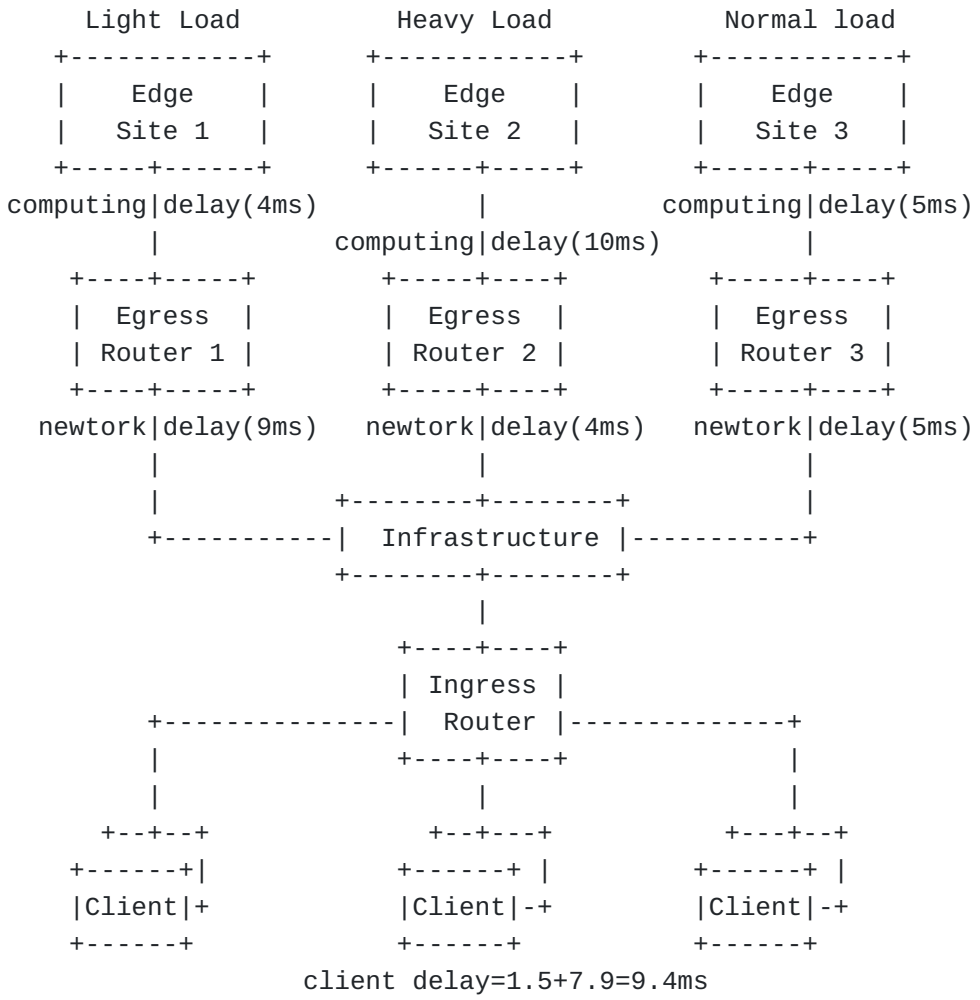

Figure 3: Computing-Aware AR or VR

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. Therefore, the transmission capacity of the network will need to be further increased, and the collected video data need 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.

4.4. Computing-Aware SD-WAN

SD-WAN provides organizations or enterprises with centralized control over multiple sites which are network endpoints including branch offices, headquarters, data centers, clouds, and more. A enterprise may deploy their services and applications in different locations to achieve optimal performance. The traffic sent by a host will take the shortest WAN path to the closest server. However, the closet server may not be the best choice with lowest cost of network and computing resources for the host. If the path computation element can consider the computing dimension information in path computation, the best path with lowest cost can be provided.

The computing related information can be the number of vCPUs of the VM running the application/services, CPU utilization rate, usage of memory, etc.

The SD-WAN can be aware of the computing resource of applications deployed in the multiple sites and can perform the routing policy according to the information is defined as the computing-aware SD-WAN.

Many enterprises are performing the cloud migration to migrate the applications from data centers to the clouds, including public, private, and hybrid clouds. The clouds resources can be from the same provider or multiple cloud providers which have some benefits including disaster recovery, load balancing, avoiding vendor lock-in.

In such cloudification deployments SD-WAN provides enterprises with centralized control over Customer-Premises Equipments(CPEs) in branch offices and the cloudified CPEs(vCPEs) in the clouds.The CPEs connect the clients in branch offices and the application servers in clouds. The same application server in different clouds is called an application instance. Different application instances have different computing resource.

SD-WAN is aware of the computing resource of applications deployed in the clouds by vCPEs, and selects the application instance for the client to visit according to computing power and the network state of WAN.

Figure 4 below illustrates Computing-aware SD-WAN for Enterprise Cloudification.

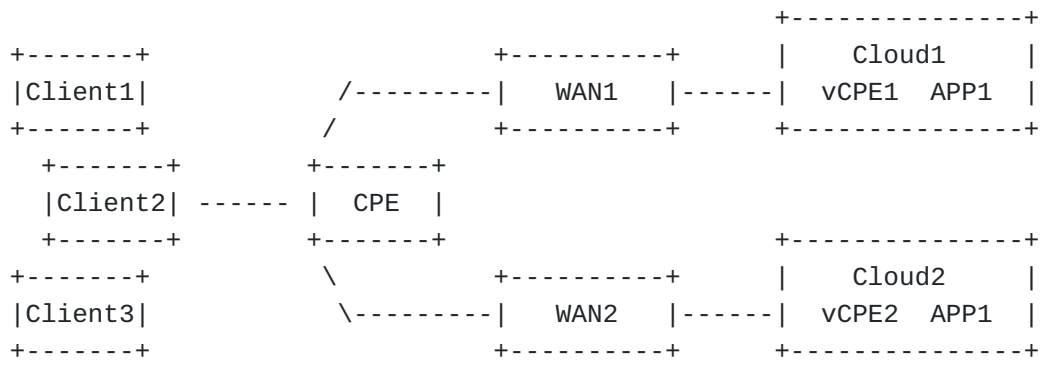

Figure 4: Illustration of Computing-aware SD-WAN for Enterprise Cloudification

The current computing load status of the application APP1 in cloud1 and cloud2 is as follows: each application uses 6 vCPUs. The load of application in cloud1 is 50%. The load of application in cloud2 is 20%. The computing resource of APP1 are collected by vCPE1 and vCPE2 respectively. Client1 and Client2 are visiting APP1 in cloud1. WAN1 and WAN2 have the same network states. Considering lightly loaded application SD-WAN selects APP1 in cloud2 for the client3 in branch office. The traffic of client3 follows the path: Client3 -> CPE -> WAN1 -> Cloud2 vCPE1 -> Cloud2 APP1

5. Requirements

In the following, we outline the requirements for the CATS system to overcome the observed problems in the realization of the use cases above.

5.1. Support dynamic and effective selection among multiple serivce instances

The basic requirement of CATS is to support the dynamic access to different service instances residing in multiple computing sites and then being aware of their status, which is also the fundamental model to enable the traffic steering and to further optimize the network and computing services. A unique service identifier is used by all the service instances for a specific service no matter which edge site an instance may attach to. The mapping of this service identifier to a network locator makes sure the data packet CATS potentially reach any of the service instances deployed in various edge sites.

Moreover, according to CATS use cases, some applications require E2E low latency, which warrants a quick mapping of the service identifier to the network locator. This leads to naturally the in-band methods, involving the consideration of using metrics that are

oriented towards compute capabilities and resources, and their corelation with services. Therefore, a desirable system

R1: MUST provide a discovery and resolving methodology for the mapping of a service identifier to a specific address.

R2: MUST provide an mapping methods for further quickly selecting the service instance.

5.2. Support Agreement on Metric Representation

Computing metrics can have many different semantics, particularly for being service- specific. Even the notion of a "computing load" metric could be represented in many different ways. 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 instance selection decision is important. Therefore, a desirable system

R3: MUST agree on using metrics that are oriented towards compute capabilities and resources and their representation among service elements in the participating edges.

R4: MUST include network metrics.

5.3. Support Moderate Metric Distributing

Network path costs in the current routing system usually do not change very frequently. Network traffic engineering metrics (such as available bandwidth) may change more frequently as traffic demands fluctuate, but distribution of these changes is normally damped so that only significant changes cause routing protocol messages.

However, metrics that are oriented towards compute capabilities and resources in general can be highly dynamic, e.g., changing rapidly with the number of sessions, the CPU/GPU utilization and the memory consumption, etc. It has to be determined at what interval or based on what events such information needs to be distributed. Overly frequent distribution with more accurate synchronization may result in unnecessary overhead in terms of signaling.

Moreover, depending on the service related decision logic, one or more metrics need to be conveyed in a CATS domain. The problem to be addressed here may be the frequency of such conveyance, thanks to the comprehensive load that a signaling process 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 considered and even be realized. Specifically, a desirable system

R5 MUST provide mechanisms to distribute the metrics

R6 MUST realize means for rate control for distributing of metrics

5.4. Support Flexible Use of Metrics

Considering computing resources assigned to a service instance on a server, which might be related to some critical metrics like the processing delay, is crucial in addition to the network delay in some cases. Therefore, the CATS components might use both the network and computing metrics for service instance selection. For this reason:

R7: a computing semantic model SHOULD be defined for the mapping selection.

We recognize that different network nodes, e.g., routers, switches, etc., may have diversified capabilities even in the same routing domain, let alone in different administrative domains. So, metrics that are oriented towards compute capabilities and resources that have been adopted by some nodes may not be supported by others, either due to technical reasons, administrative reasons, or something else. There exist scenarios in which a node supporting service-specific metrics might prefer some type of metrics to others[TR22.874]. Of course, specific metrics might not be utilized at all in other scenarios. Hence:

R8: there MUST exist flexibility in term of metrics definition and utilization for the selection of service instance.

Therefore, a desirable system

R9: MUST set up metric information that can be understood by CATS components.

R10: MUST use network and computing metrics in a flexible way that includes a default action for the interoperation of network nodes which may or may not support the specific metrics.

5.5. Support Session and Service Continuity

In the CATS system, a service may be provided by one or more service instances that would be deployed at different locations in the network. Each instance provides equivalent service functionality to their respective clients. The decision logic of the instance selection are subject to the normal packet level communication and packets are forwarded based on the operating status of both network and computing resources. This resource status will likely change over time, leading to individual packets potentially being sent to different network locations, possibly segmenting individual service

transactions and breaking service-level semantics. Moreover, when a client moves, the access point might change and successively lead to the same result of the change of service instance. If execution changes from one (e.g., virtualized) service instance to another, state/context needs transfer to another. Such required transfer of state/context makes it desirable to have session persistence (or instance affinity) as the default, removing the need for explicit context transfer, while also supporting an explicit state/context transfer (e.g., when metrics change significantly). So in those situations:

R11: session as well as service continuity MUST be maintained.

The nature of this continuity is highly dependent on the nature of the specific service, which could be seen as a 'instance affinity' to represent the relationship. 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 the AR/VR use case, where previous client input is needed to render subsequent frames.

However, a client, e.g., a mobile UE, may have many applications running. If all, or majority, of the applications request the CATS-based 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 place a heavy burden on 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.

Besides, there might be the case that UE moves to a new (access) network or the service instance is migrated to another cloud, which cause the unreachable or inconvenient of the original service instance. So the UE and service instance mobility also need to be considered.

Therefore, a desirable system

R12: MUST maintain "instance affinity" which MAY span one or more service requests, i.e., all the packets from the same application-

level flow MUST go to the same service instance unless the original service instance is unreachable

R13: MUST avoid keeping fine runtime-state granularity in network nodes for providing session and service continuity.

R14: MUST provide mechanisms to minimize client side states in order to achieve the instance affinity.

R15: SHOULD support the UE and service instance mobility.

5.6. Preserve Communication Confidentiality

Exposing the information of computing resources to the network may lead to the leakage of computing domain and application privacy. In order to prevent it, it need to consider the methods to process the sensitive information related to computing domain. For instance, using general anonymous methods, including hiding the key information representing the identification of devices, or using an index to represent the service level of computing resources, or using customized information exposure strategies according to specific application requirements or network scheduling requirements. At the same time, when anonymity is achieved, it is also necessary to consider whether the computing information exposed in the network can help make full use of traffic steering. Therefore, a CATS system

R16: MUST preserve the confidentiality of the communication relation between user and service provider by minimizing the exposure of user-relevant information according to user needs.

6. Security Considerations

CATS decision making process is deeply related to computing and network status as well as some service information. Some security issues need to be considered when designing CATS system.

Service data sometimes needs to be moved among different edge sites to maintain service consistency and availability. Therefore:

R17: service data MUST be protected from interception.

The act of making compute requests may reveal the nature of user's activities, so that:

R18: the nature of user's activities SHOULD be hidden as much as possible.

The behavior of the network can be adversely affected by modifying or interfering with advertisements of computing resource

availability. Such attacks could deprive users' of the services they desires, and might be used to divert traffic to interception points. Therefore,

R19: secure advertisements are REQUIRED to prevent rogue nodes from participating in the network.

7. IANA Considerations

This document makes no requests for IANA action.

8. Contributors

The following people have substantially contributed to this document:

Peter Willis
pjw7904@rjt.edu

Philip Eardley
philip.eardley@googlemail.com

Tianji Jiang
China Mobile
tianjijiang@chinamobile.com

Markus Amend
Deutsche Telekom
Markus.Amend@telekom.de

Guangping Huang
ZTE
huang.guangping@zte.com.cn

9. Acknowledgements

The author would like to thank Adrian Farrel, Peng Liu, Luigi IANNONE, Christian Jacquenet and Yuexia Fu for their valuable suggestions to this document.

10. References

10.1. 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>.

[RFC7285] Alimi, R., Ed., Penno, R., Ed., Yang, Y., Ed., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol", RFC 7285, DOI 10.17487/RFC7285, September 2014, <https://www.rfc-editor.org/info/rfc7285>.

[RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <https://www.rfc-editor.org/info/rfc7665>.

[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>.

10.2. Informative References

[I-D.contreras-alto-service-edge] Contreras, L. M., Randriamasy, S., Ros-Giralt, J., Perez, D. A. L., and C. E. Rothenberg, "Use of ALTO for Determining Service Edge", Work in Progress, Internet-Draft, draft-contreras-alto-service-edge-09, 10 July 2023, <https://datatracker.ietf.org/doc/html/draft-contreras-alto-service-edge-09>.

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

Authors' Addresses

Kehan Yao
China Mobile

Email: yaokehan@chinamobile.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

Hang Shi
Huawei Technologies

Email: shihang9@huawei.com

Yizhou Li
Huawei Technologies

Email: liyizhou@huawei.com

Shuai Zhang
China Unicom

Email: zhangs366@chinaunicom.cn