Workgroup: rtgwg
Internet-Draft: draft-liu-can-gap-reqs-00
Published: 23 October 2022
Intended Status: Informational
Expires: 26 April 2023
Authors: P. Liu
China Mobile
T. Jiang
China Mobile
P. Eardley
D. Trossen
Huawei Technologies
C. Li
Huawei Technologies
G. Huang
ZTE

# Computing-Aware Networking (CAN) Gap Analysis and Requirements

## Abstract

This document provides gap analysis and requirements for the problems and use cases that champion the joint optimization of both network and computing resources as outlined in[I-D.liu-can-ps-usecases]. It also identifies the 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 26 April 2023.

**Table of Contents**



## 1. Introduction

Computing service instances deployed at multiple geographically distributed edge sites are used to better realize an edge computing service in Computing-Aware Networking(CAN) use cases, as shown in [I-D.liu-can-ps-usecases]. A fundamental requirement in this type of deployment is to optimally deliver a service request to the most appropriate service instance, which would be dynamically selected by taking into consideration both the available computing resources and the quality of various network paths. Moreover, the potential requirement of the service & session continuity for a client transaction over its lifetime, possibly consisting of multiple requests, suggests some mechanism(s) be in place to maintain the

service affinity between the client and the dynamically chosen service instance.

Overall, traditional techniques to manage the load distribution or balancing of clients requests include either the choose-the-closest or the round- robin mode. Solutions derived from these techniques are relatively static, which may lead to an unbalanced distribution in terms of network utilization and computational load among available resources. For example, DNS-based load balancing usually configures a domain in the Domain Name System (DNS) such that client requests to that domain name will be statically resolved to one of several pre-provisioned IP addresses, with each IP corresponding to one node out of a group of servers. Successively, the client loads are distributed to the selected server, without further considering the dynamism of the server environment.

Certainly, there do exist some dynamic solutions to distribute client requests to servers that best fit somewhat service-specific metrics, such as the best available resources, the most powerful GPUs, the minimal platform load, and so on. These solutions usually involve the Layer 4 - Layer 7 handling of packets, such as through DNS-based or indirection servers. Unfortunately, this category of approaches is inefficient for large number of short connections. Another disadvantage (of the approaches) falls in their lacking of effective ways to retrieve the desired metrics, such as the runtime status of network devices, in a real-time way. 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 specific services that have multiple service instances residing at multiple edges, while taking into account both computing and service-specific metrics in the distribution decision, is seen as a problem of dynamically dispatching service requests, without prescribing the use of a routing solution.

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

As the use cases in [I-D.liu-can-ps-usecases] , for some applications and in some use case, considering both the computing resource and network resource status to make the traffic steering

decisions is necessary to meet the demands of latency. Before that, those status could be collocted and then be used together. This draft provides the requirements to realize the potential Computing-Aware Networking by addressing the challenges as demonstrated by typical use cases in CAN from the perspective of network that is aware of the computing resource status.

## 2. Definition of Terms

**Computing-Aware Networking(CAN):** Aiming at computing and network resource optimization by being aware of not only routing metric but also computing resource metric in deploying computing and network resource, steering traffic to appropriate computing resources, etc.

**CAN Components:** The network devices and functions that could realize CAN's demands & objectives.

**Service:** A monolithic functionality that is provided by an endpoint according to the specification for said service. A composite service can be built by orchestrating monolithic services.

**Service instance:** Running environment (e.g., a node) that makes the functionality of a service available. One service can have several instances running at different network locations.

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

**Service transaction:** Has one or more several service request that has several flows which require the affinity because of the transaction related state.

**Instance affinity:** To maintain the request of several flows belongs to the same service transaction to the same service instance.

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

## 3. Gap Analysis of Existing Solutions

There are a number of problems that may occur when realizing the use cases based on existing solutions. This section analyzes the gap of

DNS, load balancer, etc. and suggests a classification for those problems to aid the possible identification of solution components for addressing them.

### 3.1. Gap Analysis of DNS and GSLB

DNS [RFC1035] uses 'early binding' to explicitly bind from the service identification to a network address. It uses 'geographical location' to pick up the closest candidate and applies 'health check' to preventing the single point failure and also realizing load balance.

Computing resource information may be collected by DNS servers for some static use cases, such as computing resource deployment. But it can not meet the use cases that needs to update or adjust frequently.

For the Early binding, clients resolve IP address first and then steer traffic accordingly to the selected edge site. Not surprisingly, most of the time, a cached copy at the client side will be used. The consequence is that sometimes stale info obtained a couple of minutes ago could be used, which makes almost impractical choose the appropriate edge site. Further, it is fairly common that a resolver and a Load Balancer (or LB) are separate entities. The incurred signaling flow between them introduces additional overhead to the decision making procedure that is comprised of sequentially resolving first and redirecting to LB second. What's more, an IP resolution is normally at the Layer 7 and being a less-efficient app-level decision process, e.g., the database lookup that is originally intended for control but not data plane speed!

For the Health check, it is designed based on infrequent periodicity with the checking interval more than 1 second. This for sure will lead to slow or not-timely switching over upon failure. On the other aspect, limited computing resources at edges render it definitely cost-prohibitive to set up any more frequent health check.

Moreover, a Load Balancer at edge usually focuses on server load to select the 'optimal' server node first (could be virtual), and then adopts the lowest-latency (or lowest-cost) routing to reach the selected server (via IP address). Obviously, this type of standalone sequential steps lacks the organic way to combine and then jointly consider both compute/server load & routing latency (and/or cost) for a better E2E guarantee . And the last but not least, how to obtain necessary metrics from mattered entities for decision is also critical .

There is also the DNS-SD[RFC6763] and Multi-cast DNS[RFC6762] that could be used to dicover the service, which might be extended to collect the computing information. However, in most cases, they are used in the LAN environment. They need enhanced work and improvement should we intend to apply them in a wider network. Moreover, the instance selection will be pushed back to the client but rely on decision criteria being multicast to all clients , so there is a scalability limit. The gap of client based solution could be found at Section 3.4.

Generally speaking, DNS is not designed for the computing information collection. The potential enhancement could only support the limited further usage because DNS usually takes several minutes to propagate an update while clients in our targeted scenarios require frequent resolution of binding. Unfortunately, updates to the mapping between a service identifier to a service instance address cannot be pushed quickly enough into the DNS. If DNS is enforced to meet this level of dynamism, frequent resolving of the same service name would likely lead to an overload of the system. These issues are also discussed in Section 5.4 of[I-D.sarathchandra-coin-appcentres]. Some work like CDNI[RFC7336] is also based on the DNS/HTTP redirection, which has the similar problems and may not be suitable for CAN.

### 3.2. Gap Analysis of Load Balancer

A Load balancer could be seen as the external components of a network, which is designed for and deployed in a computing domain to support balanced load distribution. It may also be based on DNS system and require app level query.

For the existing load balancer solutions, there are two common ways. One way is to deploy a single load balancer at a central location for all service instances across different sites. It is the common way and is the easiest to implement. However, it bears the risk of the single point of failure. Plus, the network path from the (centrally-located) LB to server instances at (remote) sites might not always be optimal. The second way is to deploy an individual load balancer in each site, with its scope of application only to service instances in the site. It is still relatively easy to deploy. But, its main deficiency lies in no more inter-site load balancing that could prevent the achievement of better traffic steering across sites.

While most load-balancing solutions revolve around the egress-side load dispatching, there exist other designs, especially in 5G mobile networks, that conforms to the ingress-side principle by putting distributed load balancers closer to UPFs, with either 1:1 or 1:N mapping. Thru some higher-level coordination with a centralized

load-balance controller residing in the mobile system, the distributed load balancers could help steer the traffic according to the running status of UPFs. Of course, further enhancement are needed to collect network status in order to support the joint optimization. More details will be explored to realize the solution and verify the feasibility.

Generally, to achieve the joint optimization of network and computing resources, a load balancer should also learn the network path status, which would lead to the problem of how to learn and use them in an efficient way.

### 3.3. Gap Analysis of ALTO

ALTO [RFC7285]addresses the problem of selecting the 'optimal' service instance as an off-path solution, which can be seen as an alternative way of tackling the problem space of CAN at the Application Layer. So in that respect, even if both ALTO and CAN target at the common problem, they have reached different approaches; further, they impose different needs with different assumptions on how applications and networks may interact.

The critical aspect is the signaling latency and the control plane load that a service-instance selection process may incur, in both on- and off-path solutions. This in turn may impact the frequency with which applications will query ALTO server(s), especially in the mobile system where UEs may move to different cell sites (gNBs) or even roam to different mobile networks that would trigger the switchover to different network paths.

As a result, off-path systems, e.g., ALTO, which are based on receiving replies for applications/services before traffic could be delivered, might not keep optimal or even valid after the handover. So, ALTO need more improvement, including possible extension to support multi-domain deployment, quick interaction among all involved entities (like applications, service instances, etc.), and the integration of more performance metric information into the system, etc.

### 3.4. Gap Analysis of Message Broker

Message brokers (MBs) could be used to dispatch the incoming service requests from clients to a suitable service instance, where such dispatching could be controlled by service-specific metrics, such as computing load. However, MBs will face the following adversities:

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

May lead to 'middleman' adverse effects on efficiency, specifically when it comes to additional latencies as experienced by clients due to the extra but necessary communication with the broker. This introduces the 'path stretch' compared to the possible direct path between client and service instance.

Preventing the DDoS attack would be entirely limited to the cases of service instances being hidden by the broker.

### 3.5. Gap Analysis of Client Based Solution

A solution that leaves the collection of computing and network resource and further dispatching of service requests entirely to the client itself may be possible to achieve the needed dynamism. However, it does bear some drawbacks: e.g., the individual destination, i.e., the network identifier for a service instance, must be known to the client a priori for direct service dispatching. While this may be viable for certain applications, it cannot generally scale to a large number of clients. Furthermore, there would exist undesirable reasons for clients to learn the identifiers of all available service instances in a service domain.

It may be undesirable for clients to learn all available service instance identifiers for reasons of Service Providers' being reluctant to expose their 'valuable' information to clients.

It may be undesirable for clients to learn all available network paths that could be obtained either directly from the operators' exposure or indirectly by clients' self measurement.

For scalability concern if the number of service instances and network paths are very high.

### 3.6. Summary of Gap Analysis

#### 3.6.1. Dynamicity of Relations

CAN is desired to be aware of multiple edge sites' computing resource status, to provide the further opportunity of traffic steering based on the specific routing decision. So the dynamicity of relations among the multiple edge sites or service instance is the basic attributes of the potential CAN system/functions. For the different further using, the degree of the dynamicity may be different. Especially the traffic steering demands a more frequently information collection and routing decision.

The mapping from a service identifier to a specific service instance that may execute the service request 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 the binding updates. While the viability of these solutions will generally subject to the additional latency that is being introduced by the resolution of the mapping via the said application server, the potentially higher frequencies of changing the mapping relation every a few service requests is seen as difficult to be practical.

Moreover, we can foresee scenarios in which such relationship may change so frequently that it occurs even at the level of each service request. One possible factor might be the frequently changing metrics for a decision making process, e.g., the latency and load (metrics) as reported from all mattered service instances. Further, the client mobility creates a natural & physical dynamics with the consequence that a 'better' service instances may become available, or, vice versa, the previous assignment of the client to a service instance may turn less optimal, leading to the reduced performance that could root in the increased latency.

Existing solutions exhibit limitations in providing the dynamic 'instance affinity'. These limitations are inherently embedded in the solution design that is used for the mapping between a service identifier and the address of a candidate service instance. This is particularly noticeable upon relying on an indirection point in the form of a resolution or load balancing server. These limitations may result in the static 'instance stickiness' that would span many service requests or even last for the lifetime of a client session. This is normally undesirable from the perspective of a service provider in terms of achieving the best balanced request handling across many or all possible service instances.

### 3.6.2. Efficiency

For different use case of further utilize the collected computing resource information, there will be different demand to meet the efficiency issues. If the computing resource information is used for service deployment or joint resource management, there is no critical latency demand for receive and refresh the information. If the computing resource information is used for traffic steering of service to different edge sites/service instance, it requires the real-time or near real-time information, and the frequecy of refresh also needs to be quick and depend on the applications' specific demand.

The use of external resolvers, such as application layer repositories in general, also affects the efficiency of the overall service request. Extra signaling process is required between a client and the resolver, possibly through application layer solutions that result in not only more message exchanges but also

increased latency thanks to the involvement of additional resolutions. Further, accommodating the instance affinities for a large number of short-live client sessions will exacerbate this additional signaling process and worsen the latencies, thus impacting the overall efficiency of the service transactions.

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.

#### 3.6.3. Complexity and Accuracy

As we can see from the efficiency discussion in the previous subsection, at the moment when external resolvers have succeeded in collecting the necessary information and processing them to select the edge node, the network and computing resource status may have changed already. Accordingly, any additional control decision on which service instance to choose and for which incoming service request requires careful planning in order to address the potential inefficiencies that are caused by extra latencies and path stretching, at a minimum. Additional control plane elements, such as brokers, are usually neither well nor optimally placed in relation to the data path that a 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 demands.

#### 3.6.4. Metric Exposure and Use

Some systems may use the geographical location, as deduced from an IP prefix, to pick up the closest edge. The issue here is that different edge sites may not be far apart in some field deployments, which renders it hard to deduce the geo-locations from IP addresses. Furthermore, the geo-location itself may not be the key distinguishing metric to be considered, particularly if the geographic co-location does not necessarily mean the congruency of various network topologies. Also, "geographically closer" cannot exclude those closer yet more loaded nodes, consequently leading to possibly worse performance for the end user.

Some solutions may also perform 'health checks' on an infrequent base (>1s) to reflect the service node status and switch over in service- degrading or failing situations. Health checks, however, inadequately reflect the overall computing status of a service instance. It may therefore not reflect at all the fundamental yet meaningful basis a suitable service instance will act upon, e.g.,

insufficiently using the number of ongoing sessions as the indicator of load. Infrequent checks would for sure lead to too coarse granularity to support high-accurate applications, e.g., applications requiring mobility-induced dynamics such as the Intelligent transportation scenario of Section 4.2 in[I-D.liu-can-ps-usecases].

Existing solutions lack the necessary information to make the right decisions 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 providers.

#### 3.6.5. Security

Resolution systems open up two dimensions of attacks, namely attacking the mapping system itself, and attacking the service instance directly after having been resolved. The latter is particularly critical for a service provider with significantly deployed service infrastructure. A resolved (global) IP address will not only enable a (malicious) client to directly attack the corresponding service instance, but also offer the client the opportunity to infer (over time) information about available service instances in the service infrastructure, which might nurture even wider and coordinated Denial-of-Service (DoS) attacks.

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. Requirements

In the following, we outline the requirements for the CAN system to overcome the observed problems in the realization of the use cases described in [I-D.liu-can-ps-usecases].

### 4.1. Support dynamic and effective selection among mutiple serivce instances

The basic requirement of CAN 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 can

potentially reach any of the service instances deployed in various edge sites.

Moreover, according to the use case stated in [I-D.liu-can-ps-usecases], some applications require the 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 metrics to make the selection mechanism either service-specific or category-specific, or both. Therefore, a desirable system

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

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

### 4.2. 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-specific instance selection decision is important. Therefore, a desirable system

o MUST agree on the service-specific metrics and their representation among service elements in the participating edges.

o MAY include network metrics

### 4.3. Support Moderate Metric Distributing

Network path costs in the current routing system usually do not change very frequently. However, computing load and service-specific metrics in general can be highly dynamic, e.g., changing rapidly with the number of sessions, 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 signalling.

Moreover, depending on the service-specific decision logic, one or more metrics will need to be conveyed in a CAN domain. Problems to be addressed here may be the loop avoidance of any advertisement of metrics as well as the frequency of such conveyance, thanks to the comprehensive load that a signalling process may add to the overall network traffic. While existing routing protocols may serve as a

baseline for signalling metrics, other means to convey the metrics can equally be considered and even be realized. Specifically, a desirable system

o MUST provide mechanisms to distribute the metrics

o MUST realize means for rate control for distributing of metrics

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

### 4.4. 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, as described in[I-D.liu-can-ps-usecases]. Therefore, the CAN components might use both the network and computing metrics for service instance selection. For this, 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, the service-specific metrics 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, there must exist flexibility in term of metrics definition and utilization for the selection of service instance. Therefore, a desirable system

o MUST set up metric information that can be understood by CAN components.

o 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.

### 4.5. Support Session and Service Continuity

In the CAN 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 session as well as service continuity must be maintained in those situations.

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 example in [I-D.liu-can-ps-usecases], 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 CAN-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

o 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

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

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

o Should support the UE and service instance mobility.

### 4.6. 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 CAN system

o 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.

## 5. Conclusion

As a consequence, the problem of satisfying service-specific metrics is challenging to allow for selecting the most suitable service instance among a pool of instances that are available to the service throughout the network. There are quite a number of observed problems in existing solutions. The use cases [I-D.liu-can-ps-usecases] as well as the categorization of the observed problems may start the process of determining how they are best explored within the IETF protocol suite or through suitable extensions to that protocol suite.

This document analyzes the gap of existing solutions and presents high-level requirements for CAN, where the architecture should address how to model, represent, distribute and use the resource information. How to realize appropriate instance selection and routing actions and how to assure service continuity in a dynamic

environment, based on the holistic consideration of network and computing metrics, are discussed.

## 6. Security Considerations

Section 4.6 discusses some security considerations.

## 7. IANA Considerations

No IANA action is required so far.

## 8. Contributors

The following people have substantially contributed to this document:


Peter Willis
BT

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>.

**[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>.

**[RFC6762]** Cheshire, S. and M. Krochmal, "Multicast DNS", RFC 6762, DOI 10.17487/RFC6762, February 2013, <https://www.rfc-editor.org/info/rfc6762>.

**[RFC6763]** Cheshire, S. and M. Krochmal, "DNS-Based Service Discovery", RFC 6763, DOI 10.17487/RFC6763, February 2013, <https://www.rfc-editor.org/info/rfc6763>.

**[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>.

**[RFC7336]** Peterson, L., Davie, B., and R. van Brandenburg, Ed., "Framework for Content Distribution Network

Interconnection (CDNI)", RFC 7336, DOI 10.17487/RFC7336, August 2014, <https://www.rfc-editor.org/info/rfc7336>.

**[I-D.liu-can-ps-usecases]** Liu, P., Eardley, P., Trossen, D., Boucadair, M., Contreras, L. M., Li, C., and Y. Li, "Computing-Aware Networking (CAN) Problem Statement and Use Cases", Work in Progress, Internet-Draft, draft-liu-can-ps-usecases-00, 23 October 2022, <https://datatracker.ietf.org/api/v1/doc/document/draft-liu-can-ps-usecases/>.

**[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, 26 January 2021, <https://www.ietf.org/archive/id/draft-sarathchandra-coin-appcentres-04.txt>.

**[I-D.contreras-alto-service-edge]** Luis Contreras, M., Lachos, D. A., Rothenberg, C. E., and S. Randriamasy, "Use of ALTO for Determining Service Edge", Work in Progress, Internet-Draft, draft-contreras-alto-service-edge-05, 11 July 2022, <https://www.ietf.org/archive/id/draft-contreras-alto-service-edge-05.txt>.

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

## Acknowledgements

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

## Authors' Addresses

Peng Liu
China Mobile

Email: liupengyjy@chinamobile.com

Tianji Jiang
China Mobile

Email: jiangtianji@chinamobile.com

Philip Eardley

Email: ietf.philip.eardley@gmail.com

Dirk Trossen
Huawei Technologies

Email: dirk.trossen@huawei.com

Cheng Li
Huawei Technologies

Email: c.l@huawei.com

Guangping Huang
ZTE

Email: huang.guangping@zte.com.cn