Workgroup: Network Working Group
Internet-Draft:
draft-du-computing-resource-representation-01
Published: 11 July 2022
Intended Status: Informational
Expires: 12 January 2023
Authors: Z. Du   Y. Fu
         China Mobile   China Mobile

Computing Resource Representation in Computing Aware Networking

Abstract

This document introduces the way of encoding service-specific information and the way of signaling it in the network.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Traditionally, the network can only do traffic engineering according to the network statuses. As the trend of computing and network convergence, some works are proposed for network to be aware of service information, and can make a better choice in the traffic steering accordingly. Computing Aware Networking (CAN) could steer the traffic based on both the network and computing statuses, which is considered as a mechanism for computing and network convergence.

In the traditional network architecture, the network is only responsible for delivering packets between servers and clients, and is not aware of the computing information. [I-D.liu-dyncast-ps-usecases] and [I-D.liu-dyncast-reqs] show that, when service instances are deployed at multiple geographical edge sites, CAN would achieve service equivalence and load balancing by considering both the service metrics and network metrics.

However, the method of notifying the service metrics in the network, representation of computing resources, and signaling of computing resource to the network are still uncertain, which is important for the network domain to know about the computing domain.

This document dose further explorations on the way of service metrics encoding and signaling. Some requirements about the service metric representation and signaling can be found in the document [I-D.liu-dyncast-gap-reqs].

2. Definition of Terms

This document makes use of the following terms:

Computing-Aware Networking (CAN): Aiming at computing and network resource optimization by steering traffic to appropriate computing resources considering not only routing metric but also computing resource metric and service affiliation.

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

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

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

Computing capacity: The ability of nodes with computing resource achieve specific result output through data processing, specifically including computing, communication, memory and storage capacity.

3. Requirements of Computing Resource Representation and Signaling

3.1. Requirements of Computing Resource Representation

The CAN needs to obtain the computing information of the computing resource for a service, to realize the traffic steering considering both network and computing status. As described in [I-D.liu-dyncast-reqs], the representation and encoding of computing metric is crucial, which is conveyed to CAN system to support the CAN components to act upon. The representation needs to express the capabilities of computing resources accurately, and the CAN system must agree on the service-specific metrics and their representation between service elements in the participating edges for the CAN components to act upon them.

Moreover, the computing resource representation need to consider the computing modeling as the requirements described in [I-D.liu-can-computing-resource-modeling]:

Support the representation of computing resources in multiple dimensions, including computing capacity, communication capacity, cache capacity and storage capacity.

Support the representation of the computing capacity in chip category, such as CPU, GPU, FPGA, ASIC, and in computing type, such as int calculation, float calculation and hash calculation.

3.2. Requirements of Computing Resource Signaling

The representation results of computing resources need to be exposed in the network to support the efficient utilizing of computing resources, or joint utilizing of both computing resources and network resources as describe in [I-D.liu-dyncast-reqs]. CAN aims at dynamic scenarios of which the status of computing resources may vary frequently, e.g., changing with the number of sessions, CPU/GPU utilization and memory space. More frequent distribution of more accurate synchronization of the real-time representation of computing resources may result in more overhead in terms of signaling. Thus, the signaling of computing resources needs to distribute and synchronize the real-time representation of computing resources efficiently to reduce the unnecessary signaling and meet the service requirements. The requirements contain several aspects as described below.

Support to signal various message based on the representation of computing resources.

Support to control the signaling rate, such as define at what interval or events to signal the information of computing resources.

Support to signal the updated information of computing resources.

Support to implement mechanisms for loop avoidance in signaling metrics, when necessary.

4. Representation of Computing Information

The main job of the network is to forward the packets of the users from the source to the destination, while the main job of the computing is to complete the various tasks of the users.

The network metrics include the bandwidth, latency, jitter, etc. They can describe the capabilities of the network, and are independent of the detailed realization of the underlayer technologies, such as the mode of the optical fiber, or the structure of a switch.

The computing metrics are more complex, which is hard to match the QoS/QoE. For example, if the task is the AI computing, such as the image processing, the computing resource can be measured by using FLOPS (Floating-point Operations Per Second) or TFLOPS (Tera FLOPS). However, it is more difficult to get the process time, which will be influenced by the current utilization rate of CPU, cache, and so on. Even some real-time OS or protocol are used, sometimes it will fail because of the deadlock or other mechanisms of OS.That is not to say there is any problem with the OS, but the complex environment in it. So, the service metric will consider more factors to judge the performance, and how to be used in another domain to guarantee the E2E service quality.

[I-D.liu-can-computing-resource-modeling] proposes a basic architecture of computing resource modeling, which considers the computing hardware types, computing task types, communication, cache, storage status, and uses the vector to represent the basic result of modeling. The vector could be:

a group of multiple vectors, to represent the evaluated level of computing, communication, cache, and storage capacity.

a single vector, to represent the single comprehensive level of overall capacity.

How to use the vector depends on the specific application domain. For the network, to preserve the metadata privacy of computing domain, usually, weighted or fuzzy processing methods are used.

4.1. Representation of Computing Metric

How to use the vector depends on the specific application demands. To preserve the metadata privacy of computing domain, usually, the weighted or fuzzy processing methods are used by CAN.

Based on [I-D.liu-can-computing-resource-modeling], to use the information of computing resource for network, we can use two general ways to represent them. One is to use single vector to represent the level, the other is to use a group of vectors to represent more detailed information.

4.1.1. Representing in a Single value

At one aspect, we can offer a general computing load information to the ingress nodes. As an example, we perhaps only need to three values:

one red value stands for the busy status,

one yellow value stands for relatively busy status,

one green value stands for free status.

Therefore, the ingress node only needs to consider the yellow edge sites and green edge sites when steering traffic, in which the green ones are more preferred.

4.1.2. Representing in Multiple values

At the other aspect, we can also offer detailed computing related information but also are expected to be the weighted value as described in [I-D.liu-can-computing-resource-modeling], such as computing capacity information includes chips category and computing task category, communication information, cache information and storage information.

Moreover, some additional information could also be represented if needed:

the service information deployed on edge sites, for example, Service ID,

the maximum session number that the edge sites can provide,

the current session number that the edge sites can provide,

the available computing infrastructure of the server, etc.

Those information may be optional and encoded as TLVs. A specific service may have a specific preferred set of TLVs. For example, if multiple instances have the same free status, the additional TLVs could be used to represent the computing resources. The detailed decision algorithm is out of scope of this document.

The informing of the TLVs should be service-specific and on-demand. Different services may care about or have subscribed different sets of TLVs. Besides, if an Ingress node receives any TLV that it does not support, the Ingress node can just ignore it.

4.2. Example Process of Computing Load Information

For a specific service, we can offer both a general computing load information and some more specific information about the computing. A general process about it is described as below.

Step1: The service instances are deployed in multiple edge sites. The ingress nodes of network working as the load balancing point needs to obtain the computing information. The service should have a specific SID, for example SID1, in the network, so that the ingress node can recognize and treat the service request differently according to SID.

Step2: After obtaining the computing information of a service related to ServiceID1 from multiple edge sites, the ingress nodes should record the computing information. Meanwhile, an ingress node should also be able to obtain network status, for example the latency to the egress of an edge site and record it.

Step3: An ingress node receives a packet targeted to the ServiceID1. According to the service metrics and network metrics it has recorded, the ingress node makes a decision about which edge site to use and forward the packet to the related egress. The selection method may be depended on the service. For example, it may be the one with the lowest latency among the ones that can offer the service, or the one with the best computing resource among the ones that have a latency fulfilling the service requirements, or a hybrid method.

The purpose of the procedure is to find an edge site that is relatively near to the client, and also have enough computing resource for the service. However, the edge sites that provide the service may be various, and perhaps have different computing abilities. Therefore, a load balancing method considering the computing resource is useful in this scenario.

5. Signaling of Computing Information

The target of CAN is to steer traffic considering both network and computing resource status. To meet the use case demands in [I-D.liu-dyncast-ps-usecases], an "on-path" decision is expected. For instance, the Ingress of the network works as the decision point to steer the traffic of the users. In this situation, the Ingress needs to know the computing information of the service instance, which could be behind the Egress. Among the computing information, some are relatively static, and some are dynamic. They may be delivered by using different means, and at different frequencies.

Besides of the computing resource modeling and computing resource representation, CAN should also focus on how to deliver the computing information from the Egress to the Ingress.

5.1. General Process of Informing

For the signaling of the computing information, a general process about it is described as below.

Step1: The gateway of the edge site collects the computing status information of the specific service instance or a categorized service. In some cases, there will be the controller in the edge site, which can help to collect the information and notify the gateway.

Step2: The Egress of CAN receives the service status information from the gateway of the edge site and notify the CAN ingress nodes.

In the first step, the controller or the gateway perhaps can communicate by PCE or other protocol for the controller. In the second step, the controller-based method can also be used; however, communications between the controller of the edge site and the controller of the network may be complicated and inefficient.

In the following section, we propose some potential ways to notify computing information, including the BGP extension, and others potential methods. When we are notifying that the edge sites have the service, i.e., a binding address for the service and the corresponding route to it, we can add additional computing information in its Extended Community.

5.2. BGP Method in Informing

As the informing of the computing information is for the edge network nodes, we can consider using BGP, specifically the MP-BGPRFC 4760 [RFC4760] . BGP is a gateway protocol that enables the network to exchange routing information between Autonomous Systems (AS). MP-BGP allows VPN edge nodes to exchange client information via different underlay networks (e.g., MPLS). As said before, we can add the computing information in the Extended Community.

When we notify the route for the specific service (naming as ServiceID1) whose address is an anycast address, in a BGP UPDATE message, the route can include many Path Attributes. The Extended Community is one of the Attributes defined in RFC 4360 [RFC4360].

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|             Type              |           Length              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|    Flag       |    Status     |          Sub-tlvs             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
      Figure 1. Format of the Computing Information in BGP
```

Type: TBD, for example, 0x0314.

Length: This refers to the total length in octets of the element excluding the Type and Length fields.

Flag: all zero.

Status: the first two bits are used.

Sub-tlvs: the sub-tlvs related to computing information.

One example of the Sub-tlvs is that the value of FLOPS that is widely used in the AI analysis scenarios. For some services that need a large amount of computing resources, we can also provide a general computing grade information of a server, such as large, middle, or small.

Besides the computing information, BGP can also be extended to exchange some other information for the CAN. While notifying the load of the computing, the network can also monitor the whole load balancing system. If any service becomes heavy load, i.e., all the service instances for the service are busy, the network should be able to inform potential inactive service points to join in the LB. Similarly, if any service becomes light-load, i.e. all the service instances for the service are relaxed, the network should also be able to inform one active service points to become inactive to release the resource to other services.

What needs to be considered more is the update frequency. The UPDATE message is sent when the network topology, path, or other status change, not cyclical. There should be a match mechanism of the computing status change of edge sites, considering the effectiveness for a given period of time, and preventing the overload of network caused by the notification of network status update, for instance, a set threshold.

5.3. Other Methods in Informing

The computing information can be treated similarly to the OAM (Operations, Administration and Maintenance) information in the network. Therefore, it should also be able to be carried in the OAM message with some proper extensions to current OAM mechanisms. Therefore, the load balancing point can collect network information via OAM mechanisms, and it can collect computing information via OAM mechanisms.

Some network programming mechanisms such as SRv6 can also be considered here. The computing information can be carried in some places of the IPv6 extension headers. For example, some data packets from the Egress to the Ingress can carry the computing information. The insertion of the computing information can take place on the Egress. It can be on-demand or periodically.

Besides BGP, OAM and network programming mechanisms, if needed, the CAN specific methodology of computing information notification could also be further formulated.

6. Conclusion

This document analyzes the requirements of computing representation and signaling, proposing some potential method to achieve them, which are the key functions of CAN.

7. IANA Considerations

TBD.

8. Security Considerations

TBD.

9. Acknowledgements

TBD.

10. Contributors

The following people have substantially contributed to this document:

Linda Dunbar

11. References

11.1. 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>.

[RFC4360]
Sangli, S., Tappan, D., and Y. Rekhter, "BGP Extended Communities Attribute", RFC 4360, DOI 10.17487/RFC4360, February 2006, <https://www.rfc-editor.org/info/rfc4360>.

[RFC4760] Bates, T., Chandra, R., Katz, D., and Y. Rekhter, "Multiprotocol Extensions for BGP-4", RFC 4760, DOI 10.17487/RFC4760, January 2007, <https://www.rfc-editor.org/info/rfc4760>.

11.2. Informative References

[I-D.liu-can-computing-resource-modeling] Liu, P., Du, Z., Rui, L., Li, W., Li, C., and G. Huang, "Computing Resource Modeling for CAN", Work in Progress, Internet-Draft, draft-liu-can-computing-resource-modeling-00, 11 July 2022, <https://www.ietf.org/archive/id/draft-liu-can-computing-resource-modeling-00.txt>.

[I-D.liu-dyncast-gap-reqs] Liu, P., Jiang, T., Eardley, P., Trossen, D., and C. Li, "Dynamic-Anycast (Dyncast) Gap analysis and Requirements", Work in Progress, Internet-Draft, draft-liu-dyncast-gap-reqs-00, 8 July 2022, <https://www.ietf.org/archive/id/draft-liu-dyncast-gap-reqs-00.txt>.

[I-D.liu-dyncast-ps-usecases]
Liu, P., Eardley, P., Trossen, D., Boucadair, M., Contreras, L. M., and C. Li, "Dynamic-Anycast (Dyncast) Use Cases and Problem Statement", Work in Progress, Internet-Draft, draft-liu-dyncast-ps-usecases-03, 7 March 2022, <https://www.ietf.org/archive/id/draft-liu-dyncast-ps-usecases-03.txt>.

[I-D.liu-dyncast-reqs] Liu, P., Jiang, T., Eardley, P., Trossen, D., and C. Li, "Dynamic-Anycast (Dyncast) Requirements", Work in Progress, Internet-Draft, draft-liu-dyncast-reqs-02, 7 March 2022, <https://www.ietf.org/archive/id/draft-liu-dyncast-reqs-02.txt>.

Authors' Addresses

Zongpeng Du
China Mobile
No.32 XuanWuMen West Street
Beijing
100053
China

Email: duzongpeng@foxmail.com

Yuexia Fu
China Mobile
No.32 XuanWuMen West Street
Beijing
100053
China

Email: fuyuexia@chinamobile.com