Optimizing Service Segmentation with Aggregated Computing-Aware Traffic Steering Metrics

Kiem Nguyen Trung*, Minh-Ngoc Tran*, Younghan Kim*
*School of Electronic Engineering, Soongsil University, Seoul, South Korea
Email: kiemnt@dcn.ssu.ac.kr, mipearlska1307@dcn.ssu.ac.kr, younghak@ssu.ac.kr

***Abstract*—Service segmentation refers to the decomposition of a service into smaller subtasks that can be executed sequentially or in parallel across multiple service sites, enabling emerging applications such as distributed AI workloads to meet stringent requirements. However, the existing Computing-Aware Traffic Steering (CATS) framework and its metric design are designed primarily for monolithic services and can lead to suboptimal end-to-end performance for service segmentation. To address this limitation, we propose a new CATS metric aggregation level, implemented as a Metric Aggregator that consolidates normalized subtask metrics into a unified metric, enabling a more efficient, stable, and scalable pipeline of service segmentation selection with reduced control-plane overhead. Experimental results on a P4-based testbed demonstrate that our approach significantly lowers routing setup latency compared to the baseline CATS metric selection method.**

***Index Terms*—Computing-Aware Traffic Steering, Service Chains, Software Defined Network, Service Segmentation, P4**

I. Introduction

Emerging applications such as augmented reality (AR), virtual reality (VR), and distributed AI workloads demand stringent Quality of Service (QoS) and Quality of Experience (QoE) guarantees. Edge computing has emerged as a promising paradigm to meet these requirements by placing computing resources closer to end users, thereby reducing latency and improving responsiveness. To support large-scale user demands, the future network landscape is expected to see an increasing deployment of small edge computing sites [1], [2]. For better availability and scalability, the same service may be deployed across multiple sites, and the Computing-Aware Traffic Steering (CATS) framework has been proposed to support them in distributed computing environments.

However, due to the inherent resource constraints of individual edge sites, many computationally intensive applications cannot be executed entirely within a single site. Service segmentation has therefore become an attractive deployment option to address this limitation. In this approach, a service is divided into smaller subtasks that can be executed either sequentially or in parallel, with their results aggregated to complete the service request [3]. For instance, an XR rendering service can be structured as a sequential pipeline of subtasks, such as render engine processing, engine adaptation, and rendering acceleration, each deployed at different edge sites. This distributed execution reduces the computational load on any single edge site, improves fault tolerance, and enhances service responsiveness.

Selecting an optimal pipeline of subtasks for serving user requests is crucial. Choosing the physically closest edge sites does not always yield the best performance, as factors such as resource shortages, network congestion, and long task queues can degrade service quality. However, the current CATS metrics definition is primarily for single, monolithic services and lacks explicit support for service segmentation. While these metrics can facilitate the selection of an optimal instance for each subtask independently, they may lead to suboptimal performance when applied to an entire subtask pipeline. This gap motivates the need for a more efficient solution.

In this paper, we address this gap by enhancing the CATS architecture to support service segmentation by introducing a new metric design that improves efficiency, stability, and scalability while minimizing control and protocol overhead. The remainder of the paper is organized as follows: Section II reviews the background and related work. Section III introduces our proposed solution. Section IV presents experimental validation on a P4-based testbed. Section V concludes the paper.

II. Background and Related Works

Computing-Aware Traffic Steering or CATS [4] is a concept introduced by the IETF CATS working group to address service delivery in distributed computing environments. The working group assumes the presence of multiple service instances providing the same service, deployed across one or more service sites. Therefore, CATS facilitates traffic steering toward the most suitable service instance location, taking into account the current state of both computing and network resources.

To support service instance selection, the working group has defined a standardized set of metrics [5], structured into three hierarchical levels, L0, L1, and L2, arranged in order of increasing abstraction to balance decision-making efficiency with the level of detail provided. L0 represents raw metrics, which are unprocessed measurements reported directly in their native units from underlying resources, such as CPU base and boost frequencies, core count, utilization. L1 normalizes these raw metrics into broader categories, such as computing or networking, producing unitless scores. L2 further consolidates multiple L1 metrics, or in some cases L0 metrics directly, into a single normalized score that reflects the overall performance of a service instance. These standardized metrics enable CATS

to select the most appropriate service instance for a given request.

However, emerging applications often exceed the capacity of a single service site. Service segmentation addresses this by dividing a service into smaller subtasks executed across multiple sites. Each subtask instance offers partial service functionality, and multiple instances may exist for the same subtask [3]. Prior research on service segmentation has primarily focused on service splitting strategies [6], [7] and placement optimization in Multi-access Edge Computing (MEC) environments [8], [9]. Moreover, according to the CATS metrics definition [5], their metrics are standardized and defined only for single services, enabling the selection of the optimal instance for an individual service. Therefore, when these decisions are applied independently to each subtask within a segmented service pipeline, the resulting end-to-end execution may not be optimal. This is because subtasks may be scheduled to run on resource-rich but geographically distant edge sites, leading to inefficiencies in the overall pipeline performance.

To address this gap, this paper proposes a new approach for optimal pipeline selection in segmented services. A detailed explanation of the proposed architecture is presented in the following section.

III. Proposed Solution

In this section, we present our solution. The most essential contribution of our work is the introduction of a new CATS metric aggregation level, Service Pipeline Metrics, illustrated in Figure 1. This level aggregates the Level 2 metrics of all services within a pipeline into a single pipeline-level metric through an aggregation function. This design addresses the limitations of the existing CATS metrics when applied to segmented services.

The aggregation is performed by the Metric Aggregator (Figure 2), which works alongside existing CATS components to normalize and combine subtask metrics for efficient and scalable pipeline selection. For example, in a segmented service with two pipelines, Pipeline 1 executing Task A at MEC 1 and Task B at MEC 2, and Pipeline 2 executing Task A at MEC 3 and Task B at MEC 2, the pipeline metric is obtained by combining the L2 metrics of its subtasks (e.g., Task A at MEC 1 with Task B at MEC 2 for Pipeline 1). At each service site hosting subtask instances, every subtask is associated with a metric subtask component, which computes a Level 2 metric representing the subtask's overall performance. This metric is derived from a set of normalized Level 1 metrics, including computation, communication, and task-specific performance indicators. Normally, the CATS Service Metrics Agent (C-SMA) is responsible for calculating and normalizing computation-related metrics into Level 1 form, and it could be placed inside CATS Forwarders. However, due to the resource constraints of CATS Forwarders, this placement is not recommended as it may negatively impact routing performance. To mitigate this, the C-SMA is instead

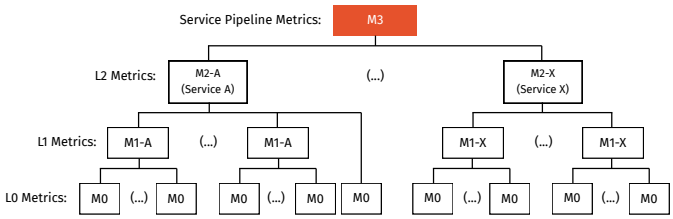

Fig. 1. New CATS Metric Aggregation Level

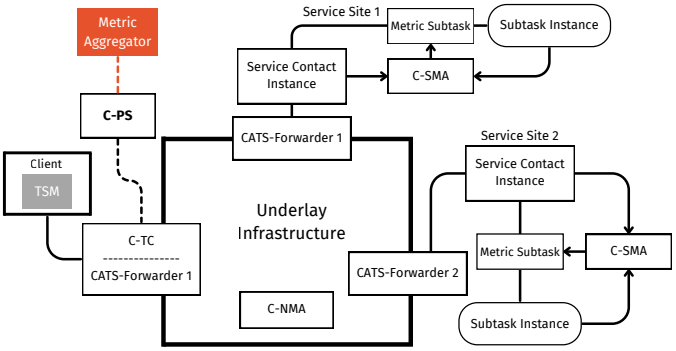

Fig. 2. Architecture of CATS framework for Segmentation Service

implemented at the service contact instance, ensuring efficient metric collection without overloading the forwarding layer.

Once all pipeline metrics are computed and normalized by the Metric Aggregator, the CATS Path Selector (C-PS) selects the optimal execution pipeline and configures the relevant CATS Forwarders with routing information. When a client issues a service segmentation request, the Task Segmentation Module (TSM) on the client side decomposes it into subtask requests, directs each subtask through the selected pipeline via the CATS Forwarders, and finally aggregates the outputs to produce the complete service result, which is returned to the client.

IV. Implementation and Result

To evaluate the proposed approach, we constructed a P4-based testbed to demonstrate its practical implementation. The experimental setup is an extended version of the testbed presented in our previous work [10]. In the evaluated scenario, a client issues a segmentation service request that is executed as a sequential pipeline of two subtasks. The first subtask performs segmentation to identify the target region, while the second subtask inserts AR effects (e.g., glasses, hats, stickers) at the correct position.

In the data plane, this segmentation service is deployed as two alternative pipelines. In Pipeline 1, the first subtask is at MEC site 1 and the second at MEC site 2; in Pipeline 2, they run at MEC sites 3 and 4 as as shown in Figure 3. Each MEC site hosts one subtask instance and its exporter process. The exporter normalizes metrics from multiple categories, such as computing, storage, obtained from the MEC node, and networking, obtained from the ONOS SDN controller and P4 switches. In the control plane, Prometheus gathers

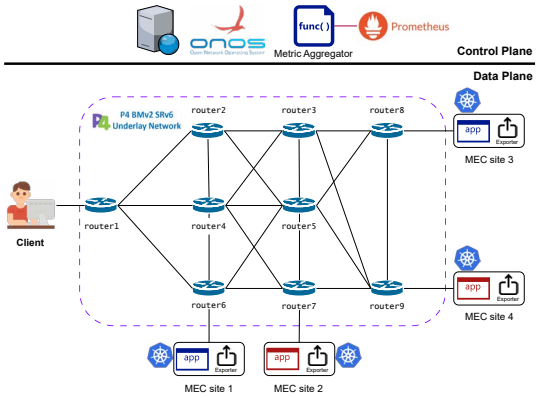

Fig. 3. Overview testbed networking

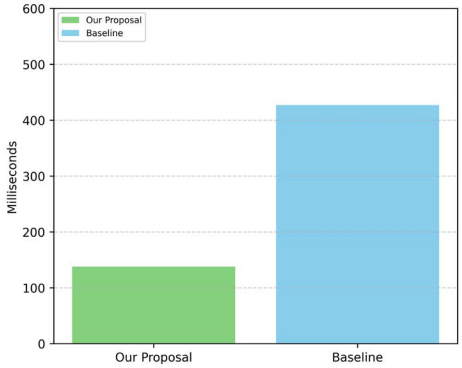

Fig. 4. Comparison between our proposal and the baseline method in setup routing time

these metrics, the Metric Aggregator normalizes them and computes a single score per pipeline, and ONOS SDN uses these aggregated metrics to select the optimal pipeline and installs routing rules to steer client traffic.

To validate the benefits of our design, we compared our proposal with a baseline method. In the baseline, all subtask metrics remain as raw metrics. In this case, ONOS must query all the relevant metrics from every MEC site and P4 switches in sequence, and perform computation to determine the optimal pipeline. This increases the control-plane overhead and routing setup time. Figure 4 presents the experimental result about the setup routing time. As expected, our approach achieves significantly lower setup latency than the baseline method. With a pipeline of two sequential subtasks, our proposed method completes the routing setup in 138 ms, while the baseline incurs much longer latency due to the additional metric collection and computation steps. The performance advantage becomes more pronounced as the number of subtasks in the service chain increases, since the baseline must query and process more raw metrics.

Overall, the results demonstrate that our aggregated pipeline-metric approach provides better efficiency, scalability, and stability compared to the baseline CATS metric selection at the instance level. Although aggregated metrics may be less precise than raw metrics, the significant reduction in overhead and routing setup latency makes them more suitable for timely decision making and direct use by network devices.

V. Conclusion

In this paper, we address the limitations of existing CATS metrics in handling service segmentation by introducing a new aggregation level, which allows efficient selection of end-to-end subtask execution pipelines. Experimental validation on a P4-based testbed demonstrated that our proposed approach reduces routing setup latency and control plane overhead compared to using raw instance-level metrics. Future work will focus on integrating into 5G/6G, particularly the Mobile User Plane domain, while also examining the precision–efficiency trade-off of aggregated metrics and extending validation to more complex service segmentation for scalability.

Acknowledgment

This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.RS-2022-II221015, Development of Candidate Element Technology for Intelligent 6G Mobile Core Network, 50) and This work was supported by Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Korea government(MSIT) (No.RS-2024-00398379, Development of High Available and High Performance 6G Cross Cloud Infrastructure Technology, 50)

References

[1] K. Nguyen Trung and Y. Kim, “Design and Implementation of a Cost-Effective Failover Mechanism for Containerized UPF,” Electronics, vol. 14, no. 15, p. 2991, Jan. 2025

[2] K. Yao, L. M. Contreras, H. Shi, S. Zhang, and Q. An, “Computing-Aware Traffic Steering (CATS) Problem Statement, Use Cases, and Requirements,” IETF, Internet Draft draft-ietf-cats-usecases-requirements-07. Accessed 15 Aug. 2025.

[3] T. M. Ngc and Y. Kim, “Additional CATS requirements consideration for Service Segmentation-related use cases,” IETF, Internet Draft draft-dcn-cats-req-service-segmentation-02. Accessed 15 Aug. 2025.

[4] C. Li, Z. Du, M. Boucadair, L. M. Contreras, and J. Drake, “A Framework for Computing-Aware Traffic Steering (CATS),” IETF, Internet Draft draft-ietf-cats-framework-11, Accessed 15 Aug. 2025.

[5] N. Li, A. Iosifidis, and Q. Zhang, “Collaborative edge computing for distributed CNN inference acceleration using receptive field-based segmentation,” Computer Networks, vol. 214, p. 109 150, Sep. 2022.

[6] C. Zhang, J. Chen, W. Li, et al., “A cloud-edge collaborative task scheduling method based on model segmentation,” Journal of Cloud Computing, vol. 13, no. 1, p. 81, Apr. 2024.

[7] M. A. Khoshkholghi, M. Gokan Khan, K. Alizadeh Noghani, et al., “Service Function Chain Placement for Joint Cost and Latency Optimization,” Mobile Networks and Applications, vol. 25, no. 6, pp. 2191–2205, Dec. 2020.

[8] B. Li, R. Yang, L. Liu, and C. Wu, “Service Place- ment and Trajectory Design for Heterogeneous Tasks in Multi-UAV Edge Computing Networks,” IEEE Internet of Things Journal, vol. 12, no. 8, pp. 9360–9371, Apr. 2025

[9] K. Yao, C. Li, L. M. Contreras, J. Ros-Giralt, and H. Shi, “CATS Metrics Definition,” IETF, Internet Draft draft-ietf-cats-metric-definition-03, Jul. 2025. Accessed 15 Aug. 2025.

[10] K. T. Nguyen and Y. Kim, “A Design and Implementation of Service Function Chaining Over Segment Routing IPv6 Network,” in 2024 15th International Conference on Information and Communication Technology Convergence (ICTC), Oct. 2024, pp. 1938–1941.