Distributed Tracing for Service Function Chaining in Named Data Networking

Haruto Kobayashi*, Kenji Kanai†, Hidenori Nakazato‡, Andrea Detti§
*Department of Computer Science and Communications Engineering, Waseda University, Tokyo, Japan, kobayashi3ch0@asagi.waseda.jp
†Waseda Research Institute for Science and Engineering, Waseda University, Tokyo, Japan
‡Faculty of Science and Engineering, Waseda University, Tokyo, Japan, nakazato@waseda.jp
§CNIT, Department of Electronic Engineering, University of Rome "Tor Vergata", Rome, Italy, andrea.detti@uniroma2.it

***Abstract*—With the rapid expansion of IoT, there is a growing demand for efficient data processing and network load optimization. Service Function Chaining (SFC) has emerged as a key technology for distributed processing, particularly in edge computing environments. While SFC is traditionally implemented over IP networks, recent research has explored Named Data Networking (NDN) as a more flexible alternative, leveraging its content caching and name-based forwarding. However, NDN's inherent request aggregation and multicast mechanisms introduce unique challenges in applying conventional distributed tracing techniques, which are crucial for SFC applications to understand potential root causes of problems.**

This paper proposes a distributed tracing method for NDN-based SFC (NDN SFC) that enables packet flow visualization and network latency analysis. Our method introduces logging agents on NDN routers to capture request flows and inter-node delays, which are then formatted using OpenTelemetry for seamless integration with visualization tools such as Grafana® and Jaeger.

Our tracing method helps to optimize service chains, to enhance network performance evaluation, and to streamline debugging, making NDN SFC deployment more practical. By offering a more efficient alternative to traditional IP-based SFC, our method supports the practical adoption of NDN SFC in IoT and edge computing environments.

***Index Terms*—Service Function Chaining, Information-Centric Networking, ICN, NDN, IoT, Distributed Tracing**

I. Introduction

With the rapid proliferation of IoT devices, there is a growing demand for technologies that efficiently process large volumes of data while optimizing network load. In particular, distributed processing techniques utilizing edge computing have gained attention, with Service Function Chaining (SFC) being one of the key approaches under consideration.

SFC is a technique for routing traffic through network services such as load balancers in a specified order, enabling optimal resource utilization. However, traditional SFC relies on fixed host-to-host routing in IP networks, limiting flexibility. Recently, Named Data Networking (NDN), a form of Information-Centric Networking (ICN), has emerged as a more adaptable approach. Unlike IP-based networking, NDN transmits data based on content names, leveraging content caching and name-based forwarding for improved efficiency.

This work was partly supported by the National Institute of Information and Communications Technology (NICT), Japan, Grant Number JPJ012368C05601.

By leveraging these properties, NDN SFC enables more dynamic and flexible service chaining compared with traditional IP-based SFC. For instance, it allows for adaptive service placement based on network load and efficient data processing in edge environments. However, realizing these benefits requires visibility into network activity and service execution, necessitating distributed tracing. A challenge is that no established tracing framework exists for NDN making network observability, while distributed tracing is widely used in IP-based networks.

In conventional IP-based SFC, distributed tracing embeds trace identifiers into packets, allowing the reconstruction of packet flows and service dependencies. However, NDN does not follow a strict request-response model. Routers can aggregate requests for the same content, and retrieved data is multicast to multiple consumers. This characteristic disrupts trace propagation, as a single identifier cannot accurately track merged or split flows. Consequently, existing tracing methods cannot be directly applied to NDN environments.

To address these challenges, we propose a distributed tracing method for NDN SFC, assuming the NDN network is implemented as an overlay on an IP network. Our method introduces external logging agents that run as separate processes on NDN routers and service nodes to capture network latency and request propagation. The collected logs are formatted using OpenTelemetry, enabling seamless integration with visualization tools such as Grafana® [1] and Jaeger [2].

By enabling integrated visualization of both service execution and packet flow, our method supports service chain optimization, performance evaluation, and debugging [3]. As previously discussed, NDN-based SFC offers greater flexibility and efficiency compared to traditional IP-based approaches. Our proposed method enhances the practical deployment of NDN SFC by visualizing both network and service behaviors, thereby supporting more effective implementations. This contributes to making NDN SFC a viable option for IoT and edge computing environments.

II. Background

A. Named Data Networking

Named Data Networking (NDN) [4] is a type of ICN that adopts name-based routing as its core mechanism. Unlike traditional host-based addressing, NDN places content at the center of networking, improving the efficiency of data delivery.

In NDN, a node that requests content is called a *Consumer*, while a node that provides content is called a *Producer*. A Consumer sends an *Interest* packet specifying the desired content name. Upon receiving this Interest, routers forward it toward the appropriate Producer based on the content name. Once the Interest reaches the Producer, the requested content is retrieved and encapsulated in a *Data* packet, which is then sent back to the Consumer along the reverse path of the Interest.

Furthermore, in NDN, Interest packets and Data packets do not always have a strict one-to-one correspondence. When multiple Consumers send Interests for the same content, routers can aggregate these requests and forward only a single Interest upstream. Once the corresponding Data is retrieved from the Producer, the router multicasts it to all requesting Consumers, reducing redundant packet transmissions. This mechanism enhances bandwidth efficiency and reduces network congestion, making NDN particularly effective for scalable content distribution.

B. NDN Function Chaining Workflow+

Several NDN-based SFC approaches have been proposed, including ICN-FC [5] and NFN [6]. NFN executes functions at a centralized router, causing scalability issues due to processing bottlenecks. ICN-FC uses ICN names to define function chains but lacks support for nested and parallel compositions.

We adopt *NDN Function Chaining Workflow+ (NDN-FCW+)* [7], which overcomes these limitations by describing service function chains using hierarchical content names. Each function autonomously retrieves and processes required data, enabling recursive execution across distributed nodes.

1) Name Structure: In NDN-FCW+, each function application is expressed as a routable NDN name, where the function prefix is followed by arguments enclosed in parentheses.

For example, applying `/A/func` to `/B/data` and `/C/data` yields:

$$/A/func/(/B/data, /C/data)$$

Note that while this syntax may appear unfamiliar, it conforms to the standard NDN name structure and can be directly used in Interest packets. This structure enables flexible composition of nested and parallel service chains using standard name-based forwarding, without additional metadata.

2) NDN-FCW+ Execution Flow: Fig. 1 illustrates the execution flow of NDN-FCW+, where the Consumer requests `/A/func/(/B/data, /C/data)`.

① The Consumer sends an Interest packet specifying `/A/func/(/B/data, /C/data)`.
② The NDN router uses longest prefix matching to forward the Interest to `/A/func`.
③ Function A parses the arguments and sends separate Interest packets for `/B/data` and `/C/data`.
④ Producer B and Producer C return Data packets containing the values of `/B/data` and `/C/data`, respectively.

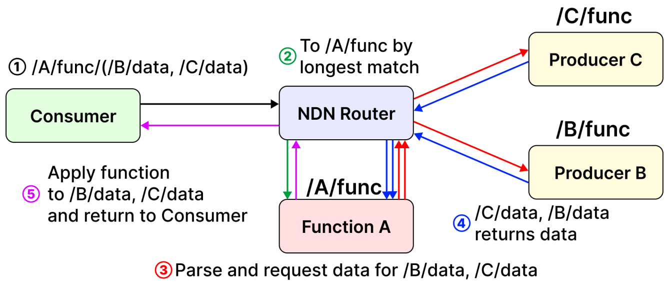

① /A/func/(/B/data, /C/data)
② To /A/func by longest match
/C/func
Producer C
Consumer
NDN Router
/B/func
Producer B
Apply function
⑤ to /B/data, /C/data
and return to Consumer
/A/func
Function A
④ /C/data, /B/data returns data
③ Parse and request data for /B/data, /C/data

Fig. 1. Execution flow of NDN-FCW+

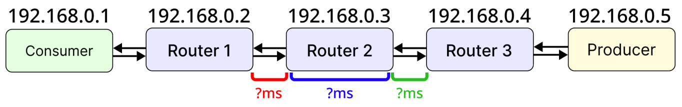


Fig. 2. Target network for tracing

⑤ Function A receives both Data packets, applies the function `/A/func`, and returns the processed result to the Consumer.

C. OpenTelemetry

OpenTelemetry [8] is an open-source observability framework for collecting, processing, and exporting telemetry data such as logs, metrics, and traces. It supports structured, end-to-end tracing using trace IDs and hierarchical spans, and integrates seamlessly with visualization tools such as Grafana and Jaeger.

This study uses OpenTelemetry to trace packet flows and service execution in an NDN SFC environment. Each span represents an event with timing and location metadata, enabling unified analysis of both network-layer delays and application-layer processing.

While ICN-native mechanisms like Reflexive Forwarding and ICN Ping provide per-hop probing, they are limited to low-level telemetry and synthetic traffic. In contrast, OpenTelemetry captures high-level dependencies across distributed nodes using actual operational requests. This enables accurate tracing of nested and parallel service chains—capabilities critical for debugging and performance optimization in SFC.

Furthermore, the OpenTelemetry ecosystem offers extensible tooling and active community support, enhancing maintainability and flexibility. These properties make it particularly suitable for observability in NDN-based SFC deployments.

1) Log Structure and Visualization: OpenTelemetry supports various log formats. In this study, we use *trace logs* to track packet flows. A trace log consists of multiple *spans*, each representing an event with a start time, an end time, and the node at which it occurred. Spans are organized in a hierarchical structure, allowing for parent-child relationships.

For instance, consider the network in Fig. 2 where a Consumer retrieves data from a Producer.

To record the transmission delays between Router1 and Router2 (red), the internal processing time at Router2 (blue), and the transmission delay between Router2 and Router3

TABLE I
SPANS TO BE RECORDED

No.	Node	Span Name	Start Time	End Time
1	router1	Forwarding to r2	2024-12-10 12:00:01	2024-12-10 12:00:02
2	router2	Processing at r2	2024-12-10 12:00:01	2024-12-10 12:00:02
3	router2	Forwarding to r3	2024-12-10 12:00:01	2024-12-10 12:00:02
4	router3	Processing at r3	2024-12-10 12:00:01	2024-12-10 12:00:02

TABLE II
SPAN RECORDING TIMINGS

No.	Node	Start Time	End Time
1	router1	Interest Sent	Data Received
2	router2	Interest Received	Data Sent
3	router2	Interest Sent	Data Received
4	router3	Interest Received	Data Sent

(green), the four spans shown in Table I are required to be recorded.

The timing of each span's recording is summarized in Table II.

Using this log format, one can calculate communication delays and processing times. For example, the delay between Router1 and Router2 corresponds to the difference between the start time of span No. 1 and the start time of span No. 2.

These spans can be exported to visualization tools such as Grafana and Jaeger via OpenTelemetry, which allows users to analyze packet traversal timing and routing structure. In our implementation, the parent-child structure of spans further enables reconstruction of the end-to-end topology of the service function chain.

2) Challenges in Applying Conventional Tracing to NDN: In OpenTelemetry, trace IDs are generally propagated by embedding them in packets, such as via HTTP traceparent headers or gRPC metadata. However, in NDN, request aggregation causes multiple requests to merge into a single Interest, complicating trace ID propagation.

When multiple Interests from different Consumers requesting the same content arrive at a router, the router aggregates them and forwards just the first received Interest upstream. Once the corresponding Data is received, it must be multicast to the requesting Consumers. Embedding a single trace ID in the Data packet cannot differentiate individual requests and it renders standard tracing approaches ineffective. Hence, an alternative mechanism is required to link requests and responses in NDN SFC.

III. Related Work

Detti et al. [9] proposed an architecture that integrates Information-Centric Networking (ICN) with Software-Defined Networking (SDN) to enable flexible management of service function chains. In their design, function chains are described using ICN names, and a centralized SDN controller orchestrates the flow of data through a sequence of named functions. While their approach successfully demonstrates function chaining over ICN, it primarily focuses on control and data delivery mechanisms. It does not address observability or provide visibility into the runtime behavior of function chains, such as per-node execution latency or service traversal paths.

Yu et al. [10] proposed a general-purpose measurement framework for NDN that supports both active and passive measurements using standard NDN primitives. Their design allows clients to request metrics such as latency and bandwidth via a modular architecture of clients, agents, and probes. While this provides a flexible mechanism for network-level diagnostics, the framework focuses on per-packet or per-link measurement operations. It does not support tracing high-level service workflows across multiple function nodes, nor does it offer integrated visualization of end-to-end Service Function Chains.

In contrast to these existing works, our proposed system enables distributed tracing and visualization of NDN-based service function chains. By introducing external log agents and employing the OpenTelemetry framework, our method provides unified observability across both application-layer function execution and network-layer routing. Furthermore, our system does not require any modification to existing NDN forwarders (e.g., NFD), ensuring non-intrusive deployment. To the best of our knowledge, this is the first approach that enables end-to-end visualization of SFC execution in NDN through distributed, per-request tracing.

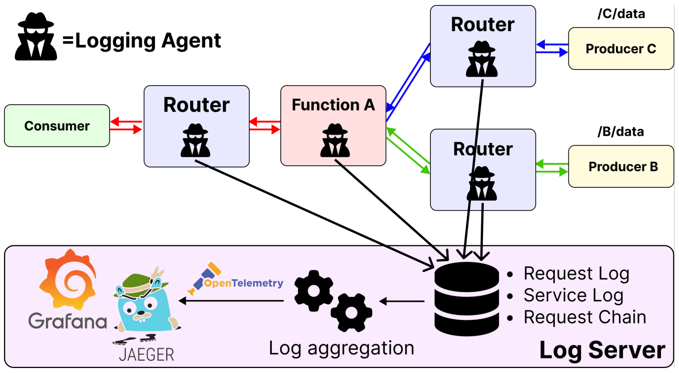


Fig. 3. Overview of the proposed method

IV. Proposed Method

This study proposes a method for tracing and visualizing the request-level packet flow and service function processing time in an SFC implemented on NDN over IP. We introduce logging agents deployed on NDN routers and service function nodes to collect transmission and reception logs of Interest and Data packets. These logs are then converted into OpenTelemetry format, enabling comprehensive network analysis using visualization tools such as Grafana and Jaeger. The overall architecture of our proposed method is illustrated in Fig. 3.

As shown in Fig. 3, logging agents deployed on routers and service function nodes collect packet transmission and reception logs and send them to a log server. The log server stores these logs in a database, aggregates the collected data, and converts them into OpenTelemetry format for visualization.

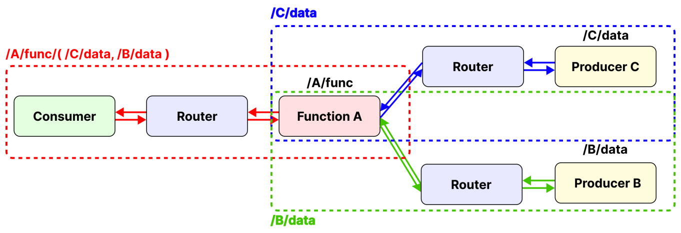


Fig. 4. Structure of a Service Function Chain in NDN-FCW+

A. Structure of Service Function Chain

In NDN-FCW+, a single SFC consists of multiple Interest requests. Each service function within the SFC generates new Interest packets to acquire necessary data, processes the obtained data, and then returns the final result to the requester. For example, in the same SFC example used in Section II-B2, function `/A/func` generates two Interest packets to request `/B/data` and `/C/data` upon the reception of request `/A/func/(/B/data, /C/data)` as shown in Fig. 4.

In NDN-FCW+, a single SFC comprises multiple Interest/Data exchanges. Therefore, to fully trace the SFC, it is essential not only to track individual requests but also to capture the relationships among them. Our proposed method records the trace IDs of each request and maintains their dependencies, enabling the reconstruction of the entire SFC structure.

B. Request-Level Packet Flow Tracing

1) Basic Concept: To enable request-level tracing without modifying NDN forwarders (e.g., NFD), our method introduces external logging agents running as separate processes on the same host as each router. These agents monitor the transmission and reception of Interest and Data packets, logging them to a central server for analysis.

Each agent maintains the following metadata:

- Content Name (e.g., `/A/func/(/B/data,/C/data)`)
- Trace ID (reused from the Interest packet's Nonce)
- Source/Destination IP Addresses
- Reception and Transmission Timestamps

These logs are used to reconstruct packet flows and compute inter-node delays, with all logic and data management handled externally, ensuring non-intrusive deployment.

2) Mapping Interest and Data Packets: As discussed in Section II-C2, NDN's request aggregation makes it difficult to trace Interest-Data relationships using embedded identifiers. To address this, our logging agents maintain *Incoming* and *Outgoing Tables* for each router, mapping Interest and Data packets based on IP addresses, content names, and timing.

The Incoming Table records received Interest packets, while the Outgoing Table stores forwarded ones. When a Data packet is received or sent, the agent uses these tables to identify the corresponding Interest and its Trace ID.

TABLE III
INCOMING TABLE (INTEREST PACKET RECEPTION)

Trace ID	Content Name	Src IP	Reception Time
12345	/A/func/(/B/data,/C/data)	192.168.1.2	2024-12-10 12:00:01
67890	/B/data	192.168.1.3	2024-12-10 12:00:02
13579	/C/data	192.168.1.4	2024-12-10 12:00:03
14879	/A/func/(/B/data,/C/data)	192.168.1.5	2024-12-10 12:00:05

TABLE IV
OUTGOING TABLE (INTEREST PACKET TRANSMISSION)

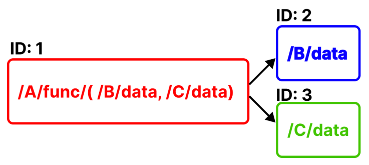
Trace ID	Content Name	Dst IP	Transmission Time
12345	/A/func/(/B/data,/c/data)	192.168.1.5	2024-12-10 12:00:01
67890	/B/data	192.168.1.6	2024-12-10 12:00:02
13579	/C/data	192.168.1.7	2024-12-10 12:00:03

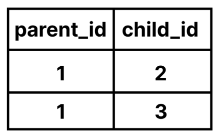
Tables III and IV show examples of the stored entries. This mechanism enables accurate tracing even in the presence of Interest aggregation, all without modifying router internals.

Each router's logging agent records the information into the Incoming Table when receiving an Interest packet and into the Outgoing Table when forwarding an Interest packet. The Nonce field in the Interest packet is used as Trace ID. When a Data packet arrives, the logging agent references the tables to associate it with the correct Trace ID and Interest by matching the IP addresses and Content Name of the Data packet with the table entries. The logging agent performs the following operations:

- Upon Interest Reception: Add an entry to Incoming Table and send a log to express Interest reception with Trace ID, Content Name, source and destination IP addresses of the Interest, and the reception time of the Interest.
- Upon Interest Transmission: Add an entry to Outgoing Table and send a log to express Interest transmission with Trace ID, Content Name, source and destination IP addresses of the Interest, and the transmission time of the Interest.
- Upon Data Reception: Retrieve Trace ID by referring to the Outgoing Table, where the source IP and Content Name of the received Data match with Dst IP and Content Name in the table, respectively. Send a log to express Data reception with the retrieved Trace ID, Content Name, source and destination IP addresses of the Data, and the time of reception.
- Upon Data Transmission: Retrieve Trace ID and Reception Time by referring to the Incoming Table, where the destination IP and Content Name of the transmitting Data match with Src IP and Content Name in the table, respectively. Send a log to express Data transmission with the retrieved Trace ID, Content Name, source and destination IP addresses of the Data, and the time of transmission.

3) Operation Example in Request Aggregation: The proposed mechanism effectively avoids Trace ID mismatches caused by request aggregation in NDN and ensures that each Interest and Data pair is correctly mapped.





parent_id	child_id
1	2
1	3

Fig. 5. SFC tracing image (Left: Chain structure, Right: Dependency table example)

In the Incoming Table shown in Table III, two Interest packets with Trace IDs `12345` and `14879` request the same content. However, due to aggregation, the router forwards only a single Interest packet upstream as shown in the Outgoing Table (Table IV).

When the Data packet with its Content Name `/A/func/(/B/data, /C/data)` arrives at the router, a log is generated as described in Section IV-B2. The router transmits two IP packets encapsulating the Data packet with two different destination IP addresses, `192.168.1.2` and `192.168.1.5`. Creating two packets out of one Data packet is the function of NDN routers if two Interests are aggregated at the routers. The two IP addresses correspond to the source IP addresses of Interest packets with Content Name `/A/func/(/B/data, /C/data)` received in advance, and are stored as Src IPs in the Incoming Table. Trace IDs, `12345` and `14879`, for the two IP packets can be found in the Incoming Table by matching the destination IP addresses and Content Names of the IP packets with Src IP and Content Name in the Incoming Table. Two logs for Data transmission with found Trace IDs are generated as described in Section IV-B2.

4) Handling of Segment Splitting: In NDN, large content is divided into multiple segments, each transmitted separately via Interest and Data packets. This study focuses on measuring the overall delay from request initiation to final data retrieval, rather than tracking each segment.

To achieve this, only the first Interest and last Data packets are recorded. The first Interest packet is identified as the request containing the base content name without a segment ID. The last Data packet is determined using the *Final Block ID*, a metadata field that indicates the highest segment number in a segmented transfer. A Data packet is recognized as the last if its segment ID matches this value.

By recording only these two packets, the method eliminates unnecessary segment-level logging while ensuring a complete evaluation of the service function chain.

C. Capturing Dependencies in Service Function Chain

As shown on the left side of Fig. 5, a single SFC consists of multiple requests. For example, a function node may send new Interest packets to fetch data from separate Producers, process the retrieved data, and then return the result to the Consumer. To reconstruct these dependencies, the function node records the Trace ID of the received Interest and the Trace ID of the newly sent Interest on the log server. This information is stored in a table, as shown on the right side of Fig. 5.

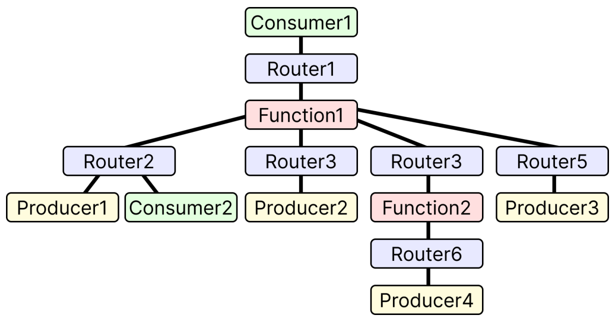


Fig. 6. Network topology used in the evaluation

D. Reconstructing the Service Function Chain

The log server stores the received packet logs in a database and reconstructs the complete SFC trace using the following steps:

1) Tracing Related Trace IDs: Starting from the Trace ID of the top-level request (the Consumer's request), the system recursively follows the dependency table created in Section IV-C to retrieve all related Trace IDs.
2) Retrieving Log Data: The system retrieves all Interest and Data packet logs corresponding to the extracted Trace IDs.
3) Reconstructing the Packet Flow: Using timestamp information, the system reconstructs the sequence of transmissions, clarifying network delays and packet routes.

E. Visualizing Using OpenTelemetry

Reconstructed trace data is converted into OpenTelemetry spans and transmitted to visualization tools like Grafana and Jaeger, enabling integrated analysis of Interest/Data transactions and service execution in an SFC.

These capabilities facilitate efficient optimization and debugging of NDN SFC deployments in edge computing and IoT environments.

V. Preliminary Evaluation

We conducted a preliminary evaluation by implementing the proposed distributed tracing mechanism in a Docker-based emulated environment, where each NDN node (including routers, function nodes, and the logging server) was deployed as an independent container. Fig. 6 shows the network topology used.

We issued the following two requests:

1) `/function1/service/(/producer1/data, /producer2/data)`
2) `/function1/service/(/function2/service/(/producer4/data), /producer3/data)`

This evaluation aims to verify the correctness of tracing and the feasibility of visualizing service execution paths and timing information. Although precise performance metrics such as delay or overhead were not measured, the system successfully

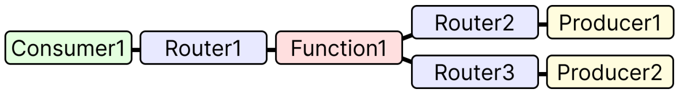


Fig. 7. Request 1) node traversal path

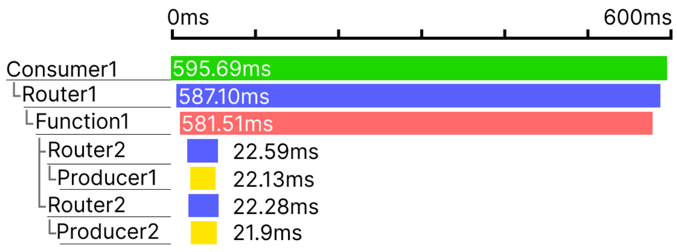


Fig. 8. Request 1) execution timeline and per-node latency

produced meaningful outputs including traversed paths and execution timeline.

Figures 7 and 8 show the reconstructed packet flow and execution timeline for request 1). Due to space constraints, we illustrate only the results of request 1), but the same tracing and visualization procedures were successfully applied to request 2) as well, confirming that the system supports nested and parallel function chaining scenarios.

Furthermore, from the logs of both request 1) and request 2), we successfully reconstructed the full SFC network topology. This included not only the paths between nodes, but also inter-node communication delays and per-node processing times, enabling comprehensive visualization of the entire execution structure.

VI. Summary

In this paper, we proposed a distributed tracing method for SFC in NDN, addressing its unique challenges and enhancing observability. Our approach enables comprehensive performance analysis and debugging by integrating service execution and packet flow tracing. The key contributions are:

1) **NDN-aware tracing mechanism**
   We handle request aggregation, multicast data delivery, and segmentation, overcoming limitations of conventional tracing techniques.
2) **Unified application and network layer visualization**
   Our method captures both service execution and packet flow, enabling detailed performance analysis.
3) **OpenTelemetry integration**
   Standardizing tracing data in OpenTelemetry format allows seamless use with tools like Grafana and Jaeger.
4) **Non-intrusive deployment**
   Logging agents can be deployed on existing NDN routers and service nodes without modifying packet structures.

Our approach improves NDN SFC's practicality by facilitating debugging, optimization, and performance evaluation. This strengthens its viability as a scalable alternative to IP-based SFC, supporting adoption in IoT and edge computing environments where flexibility and efficiency are critical.

The full source code used in this study, including the implementation of logging agents and the tracing visualization pipeline, is publicly available at the following GitHub repository: https://github.com/kobayashiharuto/NDN-FC-WorkflowPlus-Tracing

VII. Future Prospects

This study demonstrated the feasibility of visualizing both network- and application-layer behaviors in NDN SFC, providing a foundation for further development and deployment.

To enhance robustness, future research could explore incorporating error handling and Interest retransmission mechanisms, leveraging OpenTelemetry's error logging for failure analysis and recovery. For scalability in large-scale environments, selective logging techniques such as prefix-based request tagging may also be necessary to reduce overhead.

Furthermore, evaluating the resource impact of logging infrastructure—such as CPU, memory, and storage usage at logging agents and servers—would be valuable for understanding its applicability under high-throughput conditions. This could inform the design of scalable architectures based on distributed log aggregation or load balancing.

Such advancements are expected to support the practical and efficient deployment of NDN SFC in edge and cloud environments.

References

[1] Grafana Labs, "Grafana Documentation", https://grafana.com/docs/grafana/latest/, Accessed: Feb. 26, 2025.

[2] Jaeger Project, "Jaeger Documentation", https://www.jaegertracing.io/docs/latest/, Accessed: Feb. 26, 2025.

[3] M. Usman, S. Ferlin, A. Brunstrom, and J. Taheri, "A Survey on Observability of Distributed Edge & Container-Based Microservices," *IEEE Access*, vol. 10, pp. 86904–86919, Jul. 2022.

[4] V. Jacobson, D.K. Smetters, J.D. Thornton, M.F. Plass, N. Briggs, and R. Braynard, "Networking named content", *Commun. ACM*, vol. 55, no. 1, pp. 117–124, 2012.

[5] Lei Liu; Yang Peng; Bahrami, Mehdi; Liguang Xie; Ito, Akira; Mnatsakanyan, Sevak; Gang Qu; Zilong Ye; Huiping Guo, "ICN-FC: An Information-Centric Networking based framework for efficient functional chaining" 2017 IEEE International Conference on Communications (ICC), pp. 1–7, 2017.

[6] Tschudin, Christian; Sifalakis, Manolis, "Named functions and cached computations" 2014 IEEE 11th Consumer Communications and Networking Conference (CCNC), pp. 851–857, 2014.

[7] H. Kobayashi and H. Nakazato, "Service Function Chain Description Scheme in Named Data Networking", in *Proc. IEEE CQR*, Sep. 2024, pp. 1–6.

[8] OpenTelemetry, "Documentation", https://opentelemetry.io/docs/, Accessed: Feb. 10, 2025.

[9] A. Detti, C. Pisa, N. Blefari-Melazzi, and M. Pomposini, "Exploiting ICN for flexible management of software-defined networks", *Proc. of the 1st ACM Conference on Information-Centric Networking (ICN)*, pp. 107–116, 2014.

[10] Y. Yu, D. Pesavento, T. Song, C. Huang, J. Burke, and L. Zhang, "A network measurement framework for named data networks", *Proc. of the 4th ACM Conference on Information-Centric Networking (ICN)*, pp. 130–140, 2017.