社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

NDN で実装した SFC における分散トレーシング手法の提案

小林　春斗†　　中里　秀則†

† 早稲田大学大学院基幹理工学研究科　〒 169–0072 東京都新宿区大久保 3–14–9
E-mail: †kobayashi3ch0@asagi.waseda.jp, ††nakazato@waseda.jp

あらまし　近年、IoT が様々な場所で活用されており、より効率的で導入が容易な IoT ネットワークの構築が求められている。この効率化を実現するための方法としてエッジコンピューティングの利用が進んでおり、特に Service Function Chaining (SFC) を Named Data Networking (NDN) 上で実装する NDN SFC が提案されている。NDN SFC は、IoT データ処理の効率化やエッジ環境でのネットワーク負荷軽減において有効である一方、NDN のパケットキャッシュや集約といった特性が、既存のトレーシング技術の適用を困難にしている。本研究では、NDN SFC のパケットフローおよびノード間のネットワークレイテンシを可視化可能とする分散トレーシング手法を提案する。本手法では、ルータ上に NDN 向けのログテーブルを配置し、サービス間のリクエストの関連性を記録することで、NDN SFC のトレーシングを実現する。また、このトレースデータを OpenTelemetry 形式で記録することで、様々な可視化ツールとの連携を可能にした。これにより、NDN 上で実装された SFC の実環境におけるチェイン最適化やリソース管理の基盤を提供し、IoT やエッジコンピューティング環境での実践的な評価と効率化を支援することを目指す。
キーワード　Service Function Chaining, 情報指向ネットワーク, ICN, NDN, IoT, 分散トレーシング

Distributed Tracing Method for SFC Implemented in NDN

Haruto KOBAYASHI† and Hidenori NAKAZATO†

† graduate School of Information engineering, Waseda University　Okubo 3–14–9, Sinjuku–ku, Tokyo, 169–0072 Japan
E-mail: †kobayashi3ch0@asagi.waseda.jp, ††nakazato@waseda.jp

Abstract　In recent years, IoT has been utilized in various fields, driving the demand for the construction of more efficient and easily deployable IoT networks. Utilizing edge computing has become a common method to enhance efficiency. In particular, the implementation of Service Function Chaining (SFC) on Named Data Networking (NDN), known as NDN SFC was proposed for edge computing for IoT. While NDN SFC offers significant potential in improving IoT data processing and reducing network loads in edge environments, the inherent characteristics of NDN, such as packet caching and aggregation, pose challenges to applying conventional tracing techniques. In this study, we propose a distributed tracing method that enables the visualization of packet flows and network latencies between nodes in NDN SFC. The proposed method introduces NDN-specific logging tables on routers and relational tables to record service request dependencies, allowing effective tracing in NDN environments. Furthermore, the logging is handled within the OpenTelemetry framework, enabling seamless integration with various visualization tools. This approach provides a foundational framework for optimizing service chains and managing resources in real-world NDN SFC implementations, thereby supporting practical evaluations and enhancements in IoT and edge computing environments.
Key words　Service Function Chaining, Information Centric Networking, ICN, NDN, IoT, Distributed Tracing

1. ま え が き

近年、IoT デバイスの急速な増加に伴い、これらのデバイスの効果的な活用と管理が重要視されている。特に、スマートシティ、自動車産業、医療分野など、幅広いアプリケーションで IoT 技術の利用が進んでおり、これにより生じる大量のデータと高速な処理が求められている。このような背景から、ネットワーク内でのデータの集約と処理を行い、パケット量を削減

する必要性が高まっている。この効率化を実現するためには、データ処理をできるだけ物理的にデバイスに近い場所で行うエッジコンピューティングの活用が望まれている。

エッジコンピューティングの実現において、Service Function Chaining (SFC) の活用が提案されている。SFC とは、ロードバランサーなどのサービスファンクションに対し適切な順番で処理要求を転送することで、計算資源やネットワーク資源を最適に活用できる手法である。この SFC を実現する方法として、近年注目されているのが Named Data Networking (NDN) を用いた実装である。NDN はコンテンツ指向ネットワークの一種であり、コンテンツ名に基づくパケットの配送を可能にする。その結果、処理要求の宛先としてファンクションが存在するホストコンピュータではなくファンクションを直接指定することが可能となり、柔軟なサービスファンクションチェインが実現できる。例えば、混雑しているファンクションを避けて別の同一のファンクションを利用することも可能である。

このような特性を活かし、NDN を用いた SFC（NDN SFC）は IoT やエッジコンピューティング実装において有望な手法である。一方で、NDN のリクエストメッセージ集約などの特性が、従来のトレーシング技術の適用を困難にしている。具体的には、リクエストとレスポンスの 1 対 1 対応が保証されないことや、配送するデータのセグメント分割が行われることでリクエスト単位でのパケットフローの追跡が難しくなる。また、既存の分散トレーシング技術はアプリケーションレイヤに重点を置いており、ネットワークレイヤの情報（どのルータを経由したか、ルータ間のレイテンシなど）を把握する手段が不足している。

本研究では、NDN 上で実装された SFC において、アプリケーションレイヤからネットワークレイヤまでリクエスト単位でトレースを可能とした。さらに、OpenTelemetry というログを扱うフレームワークを用いることで、Jaeger や Grafana Tempo など OpenTelemetry に対応している様々な可視化ツールで扱えるようにした。これにより、リクエスト単位の可視化だけではなく、これらの情報を集約してネットワーク全体を可視化したりサービスファンクションの状態を確認できるようになる。これはまた、NDN 上で実装された SFC において実環境でのチェイン最適化のためのツールやその評価基盤を提供する。これらの情報を活用すれば、エッジコンピューティングやクラウド環境での SFC の効率化や評価実験を実践的かつ容易にすることができる。

2. 関 連 研 究

2.1 Named Data Networking

Named Data Networking (NDN) [1] は、Information-Centric Networking (ICN) の一つであり、コンテンツ名に基づいてルーティングを行うネットワークアーキテクチャである。このアーキテクチャでは、従来のホストベースのアドレス指定ではなく、ネットワークの中心にコンテンツそのものを据えることで、データ配信の効率性を向上させる。

NDN において、コンテンツを要求するノードを Consumer、コンテンツを提供するノードを Producer と呼ぶ。Consumer は、要求するコンテンツ名を記載した Interest パケットを送信する。ルータはこの Interest パケットを受信すると、コンテンツ名に基づいて適切な Producer に向けてパケットを転送する。Interest パケットが Producer に到達すると、対応するコンテンツが取得され、それが Data パケットに格納されて Consumer に返送される。この Data パケットは、Interest パケットが通過したルートを逆方向にたどり、Consumer に届けられる。

2.2 NDN Function Chaining Workflow +

NDN SFC の実装はいくつか提案されているが、本研究では、NDN Function Chaining Workflow + (NDN-FCW+) [2] で実装された NDN SFC をトレーシングする。

NDN-FCW+では、コンテンツ名の記法（名前構造）でサービスファンクションチェインを表現する。また、それぞれのサービスファンクションが自身の必要なデータを自らリクエストし、返ってきたデータに処理をかけて返すという処理を再帰的に行うことで並列やネストした複雑なサービスファンクションチェインを実現する。

2.2.1 名 前 構 造

NDN-FCW+はコンテンツ名に対して適用したいファンクション名とスラッシュをプレフィックスとして括弧で囲むことでファンクションの適用を表現する。また、カンマで区切ることで複数のコンテンツを引数としてファンクションを適用できる。例えば、/B/data, /C/data のデータの 2 つを引数として/A/func を適用したい場合は図 1 のように記述される。

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

/B/data, /C/data のコンテンツを引数として /A/func の function を適用する。

図 1　複数引数をもつファンクションの表現

2.2.2 NDN-FCW+の動作

具体的な NDN-FCW+の動作例として、図 1 の記述の処理は図 2 のようになる。

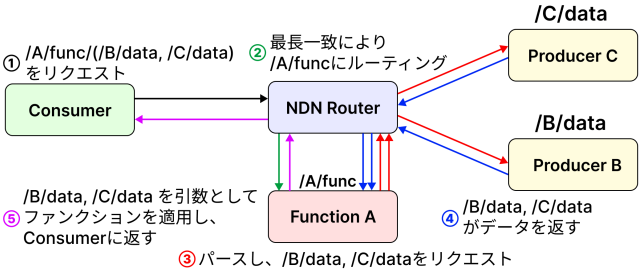


図 2　NDN-FCW+の動作

図 2 の処理は以下の通りである。

① Consumer は Interest パケットのコンテンツ名に/A/func/(/B/data, /C/data) を書き込み送信する。

② NDN Router は最長一致により/A/func にルーティングする。

③ Function A はパースを行い、必要な引数である /B/data, /C/data をリクエストする Interest パケットをそれぞれ送信する。

④ Producer B, C がそれぞれデータを返送する。

⑤ /B/data, /C/data のデータを受け取った Function A は、それらを引数としてファンクションを適用し、結果を Consumer に返送する。

3. トレーシング技術

3.1 OpenTelemetry

OpenTelemetry とはオープンソースのオブザーバビリティフレームワークである。このフレームワークは様々な可視化ツールのログ作成/管理に用いられているため、OpenTelemetry のフレームワークでログを扱うことで様々な可視化ツールの利用が可能となる。本実験ではログを作成し送信する部分で OpenTelemetry を用いる。これにより、Grafana など既存の可視化ツールを利用可能にする。

3.1.1 ログの構造と可視化

OpenTelemetry 形式で扱えるログは様々なものがあるが、本実験ではパケットをトレースしたいため、トレースログを使用する。一連のトレースログはスパンという概念で構成されている。これは"開始時間から終了時間"と"どのノードで発生したか"を記録することができ、かつスパンはツリー構造で親子関係を示せる。例として図 3 のネットワークにて、Consumer が Producer からデータを取得したときのトレースを考える。

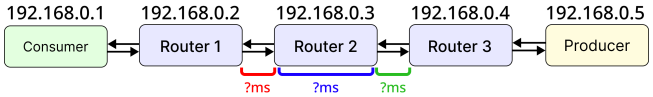


図 3 トレース対象のネットワーク

ここで Router1 と Router2 の通信時間 (図の赤)、Router2 の内部の処理時間 (図の青)、Router2 と Router3 の通信時間 (図の緑) を記録したい場合、記録するべきスパンは表 1 に示す 4 つとなる。

表 1 記録すべきスパン

No.	ノード名	スパン名	開始時間	終了時間
1	router1	fowarding /router2	2024-12-10...	2024-12-10...
2	router2	fowarding /router2	2024-12-10...	2024-12-10...
3	router2	fowarding /router3	2024-12-10...	2024-12-10...
4	router3	fowarding /router3	2024-12-10...	2024-12-10...

ここで、それぞれのスパンの記録タイミングは表 2 となる。

表 2 ノードの記録タイミング

No.	記録するノード	開始時間	終了時間
1	router1	Interest 送信時	Data 受信時
2	router2	Interest 受信時	Data 送信時
3	router2	Interest 送信時	Data 受信時
4	router3	Interest 受信時	Data 送信時

この形式のログを使用することで、通信時間や処理時間を全て求めることが可能となる。例えば Router1 から Router2 の通信時間は No.1 の開始時間と No.2 の開始時間の差である。

このスパンを OpenTelemetry フレームワークを用いて Grafana に送信すると図 4 のようなフレームグラフを確認できる。図 4 の赤の部分が Router1 と Router2 の通信時間、青の部分が Router2 の内部の処理時間、緑の部分が Router2 と Router3 の通信時間と判断出来る。

図 4 フレームグラフ

また、スパンは親子関係も示すため以下の図 5 のように構成するノードのネットワークの可視化も可能である。

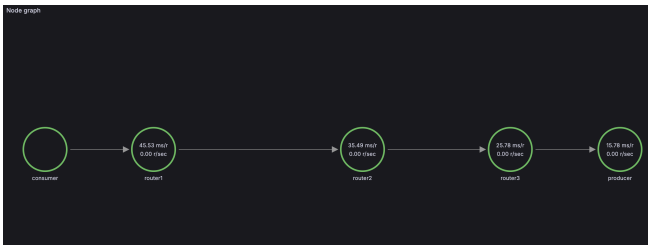


図 5 ノードのネットワーク

3.2 従来のトレーシング技術を NDN に適用する際の問題点

OpenTelemetry フレームワークでは、トレース ID をアプリケーション間で伝播することでパケットフローを追跡する。これらの伝播の方法としてパケットにトレース ID を付与している。例えば HTTP であれば traceparent ヘッダーを用いたり、gRPC のメタデータにトレース ID を付与するなどである。

しかし NDN においては、2 つ以上のリクエストでパケットが 1 つになってしまうパケットの集約というものが発生する。これは 2 つ以上のリクエストにおいて、同じコンテンツ名かつ同じインタフェースに返す状況で発生する。この状態では、パケットにトレース ID を入れていても少なくとも 1 つは消えてしまうためトレースが成立しなくなる。このように、NDN では従来のトレーシング技術で用いられているパケットにトレース ID を入れるという手法を使用することが出来ないため、別の方法を使用しなければならない。

4. 提案手法

サービスファンクションチェインにおける各リクエスト単位

でそれぞれサービスファンクションの処理時間、ルータやファンクション間の通信時間を記録する。それらを OpenTelemetry 形式に変換して可視化ツールを用いることで、3.1 節で示したようにリクエスト単位のパケットフローやネットワーク全体のレイテンシ、サービスファンクションの処理時間を可視化する。

仕組みとしてはルータとファンクションにログエージェントを配置し、別に用意したログ収集サーバにパケットログやサービス処理のログなどを送る構造となっている。ログ収集サーバでは受け取ったログをデータベースに格納する。そしてログ収集サーバではデータベースのログを集計して OpenTelemetry 形式に変換して指定した宛先に送信する機能を提供する。これにより、Grafana, Jaeger, ZipKin など OpenTelemetry に対応するあらゆる可視化サービスでログを分析可能となる。これらの構成を図 6 に示す。

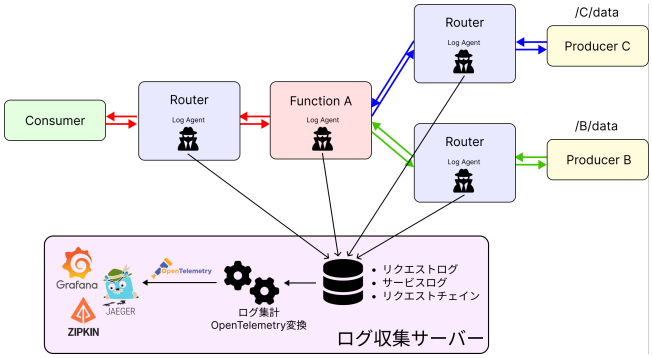


図 6　トレーシング構成

4.1　サービスファンクションチェインの構成

まずはサービスファンクションチェインの構成を示す。例えば /A/func/(/C/data, /B/data) のリクエストでは、ファンクションである /A/func が /C/data と /B/data の 2 つのリクエストをすることでデータを集め、処理した結果をこのリクエストの要求元に返すという形になる。ゆえに以下の図 7 のように、ひとつのサービスファンクションチェインが 3 つのリクエストで構成されていることとなる。

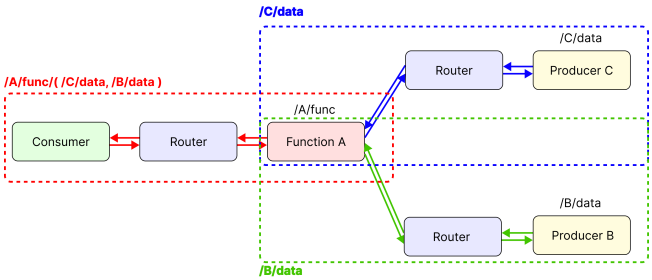


図 7　サービスファンクションチェインの構成

そのため、3 つのリクエストを 1 つのサービスファンクションチェインとして可視化するためには、まず各リクエスト単位でのパケットフローをトレースし、各リクエストがどのように繋がっているかをトレースする必要がある。

4.2　リクエスト単位のパケットフローをトレースする

まずはリクエスト単位のパケットフローをトレースする手法を示す。ルータとファンクションに配置されたログエージェントがパケットログを生成する。パケットログとして Interst パケットと Data パケットの受信時刻と送信時刻を記録し、それを集約することでリクエスト単位でのトレースが可能となる（図 8)。ただし、ただ記録するだけではどのログがどのリクエスト

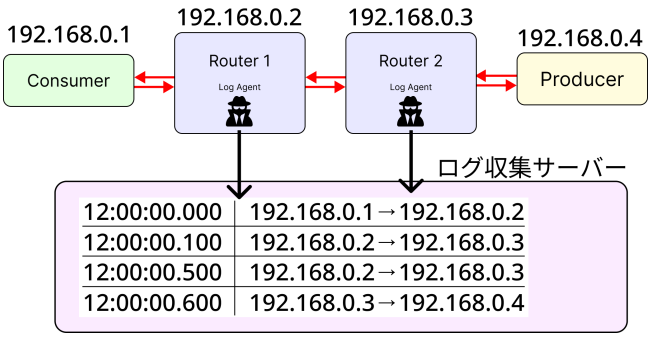


図 8　ログ収集

に紐づいているかがわからない。そのため、リクエストに ID として RequestID をつける。そして、RequestID をログデータに付与して記録する。ここで、RequestID として Interest パケットに含まれる Nonce を利用することとする。

一方、Data パケットには Nonce は存在せず、3.2 節にて述べた通りパケットが集約される可能性があるため、パケットに RequestID を埋め込むことは出来ない。

そこで、ログエージェントは、Data パケットを待機中のリクエストの Interest パケットを記録しておく。そして Data パケットが来た時に対応する Interest パケットを割り出せるようにする。具体的には、ログエージェントは Interest パケット受信時に、コンテンツ名、RequestID、送信元 IP アドレスを記録する。そして、Data パケットを受信したときに Data パケットのコンテンツ名と送信先 IP アドレスをキーとして検索する。

これを踏まえるとログエージェントの動作は以下となる。

（1） Interest パケット受信時: Interest パケットの情報もとに待機中のリクエストとして記録。そしてログ収集サーバーにコンテンツ名、RequestID、時間、送信元、送信先、受信ノードの IP アドレスを記録。

（2） Data パケット受信時: 待機中のリクエストから対応する Interset パケットを確認し、RequestID を割り出す。その RequestID をもとにログ収集サーバにコンテンツ名、RequestID、時間、送信元、送信先、受信ノードの IP アドレスを記録。

そしてこれはリクエスト集約時にも問題なく動作する。リクエストが集約されるのは、ルータ内で同じインタフェースへ返却する 2 つ以上の同じコンテンツ名のリクエストが待機しているときである。この場合、対応する待機中のリクエストは全て記録されており、Data パケット受信時に Data パケットのコンテンツ名と送信元 IP アドレスで検索するとそれぞれの対応する Interest を割り出せる。これをもとに、それぞれの RequestID でログ収集サーバに記録を行うことにより、それぞれのリクエストのトレース情報を記録できる。

4.2.1　セグメント分割への対応

NDN では、大きなコンテンツの場合、複数のセグメントに分割し、複数の Interest パケットと Data パケットで通信を行う。しかし、本手法の目的は、コンテンツのリクエスト開始からデータ取得完了までの全体的な時間を計測することである。そのため、各セグメントごとの詳細なパケットフローを追跡する必要はない。そこで、ログ収集の対象を以下の 2 つのパケットに限定する。

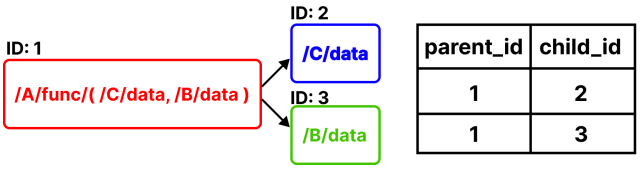
（1） 最初の Interest パケット：コンテンツのリクエスト開始時点を示す。

（2） 最後の Data パケット：コンテンツのデータ取得完了時点を示す。

この 2 つのパケットのみをログ生成の対象とすることでセグメント分割による複雑さを軽減し、効率的なログ収集と解析を可能にする。

4.3　サービスファンクションチェインのトレース

図 7 で示した通り、サービスファンクションチェイン単位でトレースするには、構成するリクエストがどのように繋がっているか記録されている必要がある。例えば図 7 の関係であれば図 9 の左に示すようなツリー構造となる。



parent_id	child_id
1	2
1	3

図 9　サービスファンクションチェインのトレース

本手法では、この関係を図 9 の右に示すテーブルでログ収集サーバーに記録する。これは、サービスファンクションが受信した Interest パケットの RequestID と、サービスファンクションから送信した新たな Interest パケットの RequestID を記録することで実現できる。これにより、一連のリクエストによるサービスファンクションチェインを構築できる。

4.4　サービスファンクションチェインの構築手順

ログ収集サーバーに記録されたログから、サービスファンクションチェインのパケットフローを構築する手順は以下となる。

（1） **関連する RequestID の収集**: トレース対象の RequestID を起点に、構成するリクエストを再帰的に参照し、関連するすべての RequestID を取得する。

（2） **ログデータの取得**: 取得したすべての RequestID に対して、ログ集約サーバーから対応する Interest パケットと Data パケットのログを収集する。

（3） **パケットフローの再構築**: 各 RequestID ごとに、ログを受信時刻順に並べる。これらを組み合わせて、サービスチェイン全体のパケットフローを再構築する。

（4） **レイテンシと処理時間の計算**: 再構築したパケットフローから、ノード間のネットワークレイテンシや各ノードの処理時間を計算する。

4.5　データの可視化

構築したサービスチェインの情報をもとに、3.1 で示した形でスパンを作成していき、OpenTelemetry 形式に変換する。

変換したログデータは Grafana などの OpenTelemetry に対応した可視化ツールに送信することで、1 つのサービスファンクションチェインのリクエストのフレームグラフや通過したノードを可視化でき、すべてのログを統合してネットワーク全体のパフォーマンスやサービスの平均処理時間なども確認可能となる。

5.　評　　価

本手法の有効性を確認するため、NDN-FCW+の環境で提案したトレーシング手法を適用する。そしてログを集計し、OpenTelemetry に変換したログデータを Grafana に送信し可視化する。今回使用するネットワーク構成は図 10 となる。

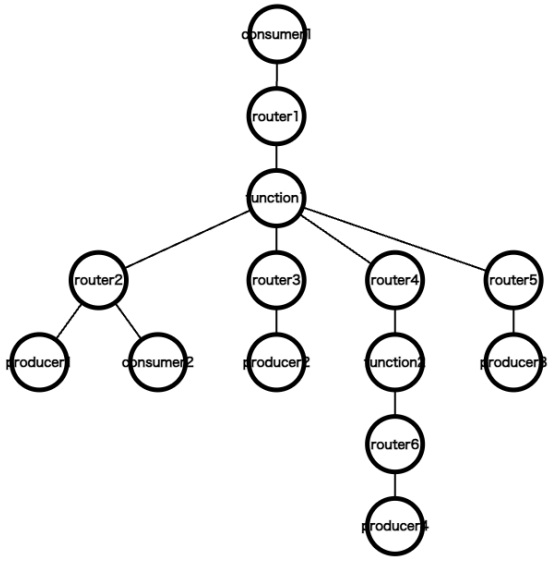


図 10　ネットワーク構成

そして、以下の二つのリクエストをそれぞれ送る。

（1） /function1/service/(/producer1/data,/producer2/data)

（2） /function1/service/
(/function2/service/(/producer4/data), /producer3/data)

そして、以下の可視化を行う。

- それぞれのサービスチェインの通過ノードの可視化
- それぞれのサービスチェインのフレームグラフ
- 全体のサービスチェインネットワークの可視化

5.1　それぞれのサービスチェインの通過ノードの可視化

Grafana の Node Graph 機能を使用し、RequestID を指定することでそれぞれのサービスチェインの通過ノードの可視化したものを図 11 と図 12 に示す。

5.2　それぞれのサービスチェインのフレームグラフ

Grafana Tempo を使用し、OpenTelemetry 形式のログからそれぞれのサービスチェインのフレームグラフの可視化したものを図 13 と図 14 に示す。それぞれのサービスは 500ms かかるように設定されている。実際にフレームグラフを見ると Producer やルーター間の遅延はほとんどなく、処理の大半がサービスの処理時間が占めていることが分かる。

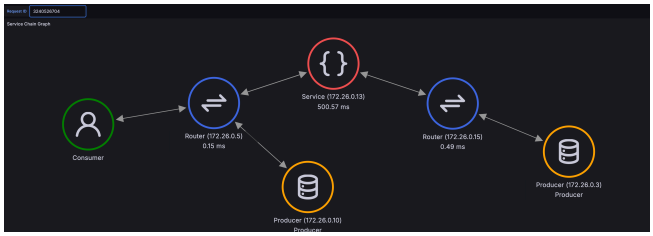


図 11　(1) のリクエストの通過ノードの可視化

図 12　(2) のリクエストの通過ノードの可視化

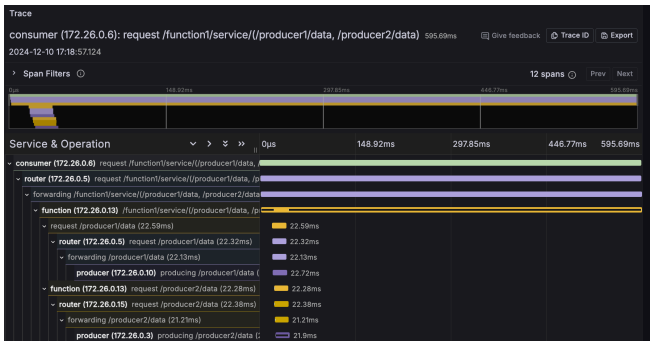


図 13　(1) のサービスチェインのフレームグラフ

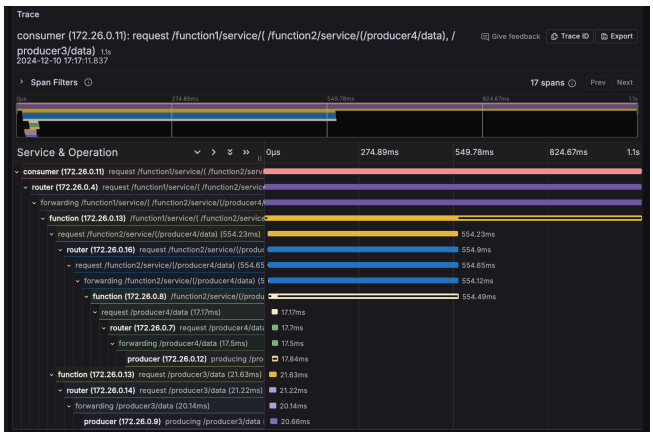


図 14　(2) のサービスチェインのフレームグラフ

5.3　全体のサービスチェインネットワークの可視化

Grafana Tempo を使用し、それぞれのサービスチェインのログから全体のサービスチェインネットワークの可視化したものを図 15 に示す。

6.　結　　論

6.1　研究結果のまとめ

本研究では、NDN 上で実装されたサービスファンクションチェイニングにおける分散トレーシング手法を提案し実装した。

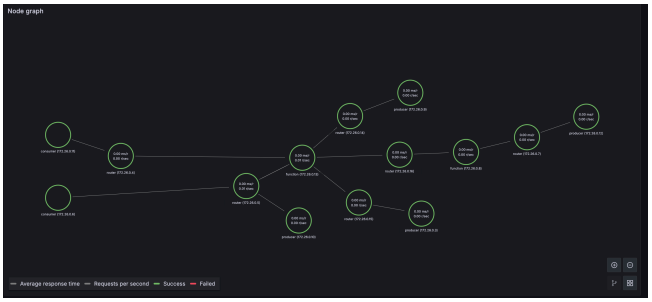


図 15　全体のサービスチェインネットワークの可視化

これは以下の重要な特徴を備えている。

（1）　**NDN 固有の課題への対応**: パケットのキャッシュや集約といった NDN 特有のトレーシングの困難さを解決した。

（2）　**アプリケーションレイヤからネットワークレイヤまでリクエスト単位でトレース可能**: 従来の分散トレーシングの手法はアプリケーションレイヤーのトレースのみであった。NDN ではネットワークレイヤーも重要なため、それに合わせてどちらのレイヤーも可視化を可能とした。

（3）　**OpenTelemetry 形式の対応**: OpenTelemetry 形式にログ変換を可能としたことで、既存の OpenTelemtry に対応する可視化ツールが利用可能となった。

（4）　**導入容易性**: パケット構造やルータの実装を変更せずに、ログエージェントを追加するだけで導入できる。

これにより、NDN SFC におけるチェインの最適化や、エッジコンピューティングやクラウド環境での効率的な SFC の実装に貢献することが期待できる。

6.2　今後の展望

現時点の実装ではエラー処理の場合の対応がなされていない。OpenTelemetry にはエラーのログを扱う方法も存在しているため、その形式に対応してエラーも扱えるようにすることでより NDN SFC の実装検証時に役に立つだろう。また、ネットワークレイヤーやサービスの可視化が出来たことで、効率的な NDN SFC を構成するためのサービスの配置戦略やルーティング戦略など、さまざまな活用が可能となる。それらの分野と組み合わせることで効率的な NDN SFC の実装が期待できる。

謝辞　本研究成果は, 国立研究開発法人情報通信研究機構の委託研究（採択番号 JPJ012368C05601）により得られたものです.

文　　献

[1] Jacobson, Van and Smetters, Diana K. and Thornton, James D. and Plass, Michael and Briggs, Nick and Braynard, Rebecca, "Networking named content", Commun. ACM, 55(1):117–124, January 2012.

[2] 小林春斗; 中里秀則, "Named Data Networking によるサービスファンクションチェインの記述法の提案", 電子情報通信学会, 2024.