一般社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
IEICE Technical Report
CAS2020-80,CS2020-87(2021-03)

NDNにおけるSFC効率化のためのキャッシュ置換ポリシー

伊藤　　快†　　中里　秀則†

† 早稲田大学基幹理工学部情報通信学科　〒169–8555 東京都新宿区大久保 3–4–1
E-mail: †konokono1117@fuji.waseda.jp, nakazato@waseda.jp

あらまし　Information Centric Networking (ICN) の実装の一つである Named Data Networking (NDN) を用いて効率の良い Service Function Chaining (SFC) を実現するためのキャッシュ置換ポリシーについて検討を行った．本稿では，NDN ルータのキャッシュを有効にした上で，SFC を行う場合を対象としている．キャッシュにおいて，ファンクション適用後のコンテンツに優先度をもたせ，キャッシュリプレイスメント発生時にリプレイスメント対象となるコンテンツとしては，ファンクションが適用されていないものを選択する手法を取り入れ，効率の良い SFC が実現可能となることを評価する．

キーワード　NDN，SFC，ファンクション選択問題，キャッシュ

Cache Replacement Policy for Service Function Chaining in NDN

Konomu ITO† and Hidenori NAKAZATO†

† Faculty of Engineering, Waseda University　Okubo 3–4–1, Shinjyuku-ku, Tokyo, 169–8555 Japan
E-mail: †konokono1117@fuji.waseda.jp, nakazato@waseda.jp

Abstract　A cache replacement policy in Service Function Chaining (SFC) in the environment of Named Data Networking (NDN), an incarnation of Information Centric Networking, is proposed. Cache is assumed to be activated in NDN router while SFC is performed in this paper. This proposal places preference on the content to which functions are applied. The content which functions are not applied is chosen when cache replacement occurs in this proposal. The proposed method was evaluated for its efficiency.

Key words　NDN, SFC, Function Selection problem, Cache

1. まえがき

近年，IoT サービスの普及から IoT デバイスの数が急増している．それに伴い通信トラヒックが増加することにより，クラウドやネットワークでの負荷が大きくなっていくことが考えられている．このことは低遅延，高信頼性を必要とする IoT サービスを運用する際には大きな課題として挙げられる．そこでこの課題の解決策として，本研究では Service Function Chaining(SFC) の導入を挙げている．SFC は IoT サービスの実現に必要な，IoT データの加工等を含めた処理，複数のデータ収集等，様々な処理を連携させることで多様な機能を迅速かつ柔軟に利用したり，ネットワークの負荷の分散を図る技術であり，本研究ではこの IoT データの様々な処理機能をファンクションと総称することとする． さらに本研究では Named Data Networking (NDN) [3] を通信プロトコルとして用いることでファンクションの割り当てを名前ベースでの宛先指定により柔軟に行う．

本研究では，ネットワーク内の各 NDN ルータのキャッシュ機能における新たなキャッシュ置換ポリシーを提案し，SFC がどの程度効率化できるかを評価した．

2. Service Function Chaining

2.1 Service Function Chaining (SFC)

End-to-End のネットワークサービスは一般的に，ファイアウォール，ロードバランサ，NAT，その他のアプリケーション特有の機能といった様々な機能 を組み合わせることでそれぞれのサービスを実現させている．SFC とはソフトウェアによって，要求されたサービスに応じてこれらの機能へ適切な順番でパケットを転送することで必要なネットワークサービスを実現させる技術である．以前は，サービスを構成する機能を専用ハードウェアで実現させており，サービスの更新や変更などにコストがかかっていた．しかし，Network Function Virtualization (NFV) という技術が登場したことにより，これらの機能を汎用ハードウェアで実現できるようになった．この NFV によって汎用サーバにソフトウェアとして実装された機能は Virtual

This article is a technical report without peer review, and its polished and/or extended version may be published elsewhere.
Copyright ©2021 by IEICE

Network Function (VNF) と呼ばれる．NFV によって，サービス要求やネットワーク状況に応じて VNF の柔軟な起動が可能となった．このように VNF の起動制御がされている中で，柔軟なパケット転送の制御が可能な SFC は効率的なネットワークサービスを実現させるにあたって非常に重要な技術である．現在，SFC 技術の標準化が IETF(Internet Engineering Task Force) によって進められており [2]，他にも SFC 技術に関連する研究がされている．

2.2 IoT サービスにおける SFC

本研究では SFC の技術を IoT サービスに適応していくことを考える．IoT サービスは様々なファンクションを組み合わせることで実現しており現在の所，一般的にはこれら機能の処理はクラウドで行っている．しかし近年の IoT デバイスの増加に伴い，データトラフィックは莫大なものとなり，IoT データに対するファンクションを全てクラウドで実行することは，クラウド，ネットワークの両者にとって非常に大きな負荷となる．そこで本研究ではこれらのファンクションをネットワーク内のルータ上にも配置することでクラウドにかかる負荷や遅延を軽減させることを考える．SFC を用いることで，ルータに配置したファンクションのうち，求める IoT サービスに必要なファンクションを経由させることでに IoT サービスを実現させる．

例として，センサなどの IoT デバイスから取得するデータを分析する IoT サービスを挙げ，図 1 に示す．ここでのファンクションとしては，「データ収集」，「データ分析」，「データ加工」と分けることとする．これらのファンクションをルータに配置し，必要なデータを SFC 技術で適切なファンクションを経由させて処理をさせることで IoT デバイスからクラウドまでのネットワーク内で必要な処理を完了させる．さらに，イベント検知などのファンクションはデータを処理した結果に応じて，クラウドまでデータを送ることなく処理を行うこともできる．

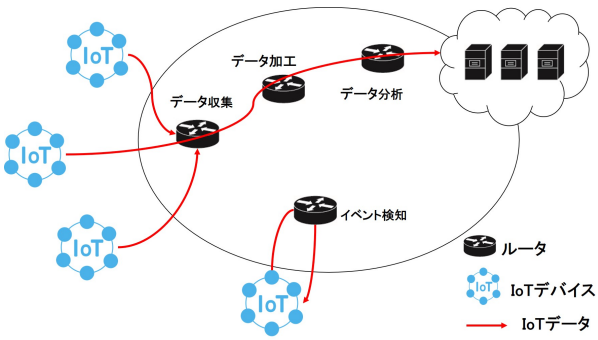

図 1　IoT サービスにおける SFC

3. NDN

3.1 NDN のアーキテクチャ

NDN における通信は，Interest パケットと Data パケットというコンテンツ名を情報として含む 2 つのパケットをユーザとルータもしくはサーバ間を送受信させることで実現させている．Interest パケットはコンテンツ要求のために用いられ，ユーザが要求するデータを識別するための名前が含まれているパケットであり，この名前に基づいて，データが存在するサーバに向かって送信される．Data パケットは目的のコンテンツがあるサーバからコンテンツ名によって要求されたデータをユーザまで送り返すパケットのことである．それぞれのパケットを図 2 に示す．

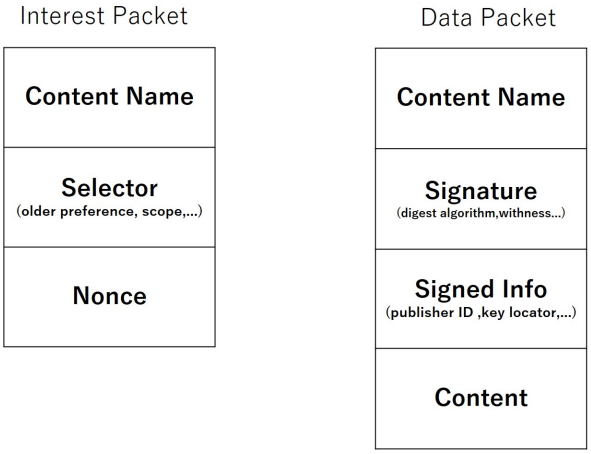

図 2　NDN における Interest パケットと Data パケット

3.2 NDN のフォワーディングプロセス

NDN では各ルータに Content Store (CS)，Forwarding Information Base (FIB)，Pending Interest Table (PIT) の 3 つのデータ構造を持たせることで Internet パケットと Data パケットのフォワーディングを行う．それぞれの構造と役割は以下のようになっている．

CS:　キャッシュ機能を実現させるデータ構造であり，Data パケットをキャッシュするためのデータストレージとなっている．CS に Data パケットが送られてくると，NDN のルータはキャッシュポリシーに基づいて Data パケットをキャッシュするかどうかを判断してパケットをキャッシュする．キャッシュする際に CS にキャッシュできる容量がない場合，キャッシュリプレイスメントポリシーに基づいてキャッシュされているパケットのどれかを破棄し，新たなパケットのキャッシュを行う．

PIT:　Data パケットが Interest パケットが通過したルータをたどれるように，Interest パケットがどのユーザ，ルータから来たかを記憶するためのデータ構造である．Interest パケットが PIT に送られてきた際に，Interest パケットに記載されているコンテンツ名に対して，Interest パケットを受け取ったインターフェイスである Face の記録を行う．Data パケットが NDN の各ルータに送られると PIT を参照し，送られてきた Data パケットに記載されているコンテンツ名に該当する PIT エントリを調べる．該当するエントリが存在した場合，この PIT エントリに記録されているフェイスに Data パケットを送信し，このエントリの破棄を行う．

FIB:　Interest パケットを目的のコンテンツ名が記載されているデータがあるサーバまで送信するための，転送先を決めるデータ構造である．NDN では Interest パケットに記されているコンテンツ名を元にフォワーディングを行うため，FIB にはコンテンツ名の prefix と次の転送先のインターフェイスであ

る Face が記載されている．Interest パケットがルータに送られると FIB を参照して，Interest パケットに記載されているコンテンツ名に最長で一致する FIB エントリを調べ，該当するエントリが存在した際，FIB エントリに記載されている Face に Interest パケットを送信する．

3.3 NDN における SFC

本研究では，[5] において提案されている NDN-FC を用いて NDN プロトコルでの SFC の実行を行うものとする．この手法では，3.1 で説明した NDN の Interest パケットに新たに Function Name と呼ばれるフィールドを追加している (図 3) [6] [5]．Function Name にはユーザーが定めた SFC リクエストに基づいたファンクション名をスラッシュで区切った階層構造で記述する．Function Name に記述したファンクション

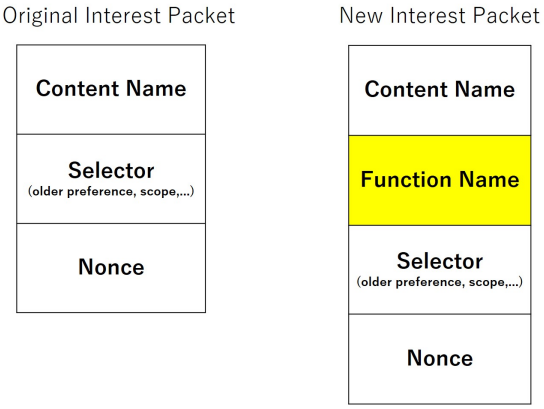

図 3　Function Name を追加した Interest パケット

名に基づきフォワーディングを行うことで SFC の実現を図っている．

以下の図 4 にネットワークのルータにファンクション F1, F2，F3 が配置された状態で F1 → F2 → F3 の順序でファンクションを実行する際のフォワーディングプロセスを示す．各ルータの FIB にはコンテンツ名と同様にファンクション名とそのフェイスが記録されており，ルータはファンクション名と FIB エントリーの最長一致により送り出すフェイスを決定する．先の例を用いると，ファンクション名の先頭のプレフィックスに F3 と記述されているため，Interest パケットを F3 が存在する場所に送り出すこととなる．Interest パケットがファンクションがある場所に届いた場合，各ファンクションは Interst パケットの Function Name を参照し，自分のファンクション名に該当する先頭のプレフィックスを削除する．すなわち，図では F3 に到着後，ファンクションヘッダは/F2/F1 と記述されていることとなる．この後再び各ルータは Function Name に基づき Interest パケットを転送し，Function Name に記述された全てのファンクションへと送られていく．この動作を繰り返し，Interest パケットが全てのファンクションを経由し，Function Name にファンクション名の記述がなくなった場合，各ルータは 3.2 で述べた NDN のフォワーディングプロセスにしたがって，コンテンツ名に基づいて Interest パケットをサーバまで送信する．Interest パケットがサーバに送り届けられた

ら，サーバは Interest パケットのコンテンツ名に該当するコンテンツの入った Data パケットを作成し，Interest パケットを受け取った Face から送出する．各ルータは PIT の情報を基に，Data パケットを Interest パケットが通った経路を逆方向に，ユーザに向けて送り返す．Data パケットがファンクションに到着したら，ファンク ションはその処理を Data パケットに加える．これを繰り返し，全てのファンクションで処理 を加えられた後，Data パケットはユーザの元へ届けられることにより SFC によって選択された一連のファンクションを実行した要求コンテンツを取得する．

本研究で使用するファンクション選択手法としては，[4] で提案された手法を用いている．この手法はファンクション呼び出し回数に着目し，ファンクション呼び出し回数が少ないファンクションも選択することで使われるファンクションの分散を図り，加えて，SFC を実行する上でパケットが経由するホップ数に着目し，ホップ数を最小限にすることでネットワーク内の遅延を抑え，SFC サービスの実行時間の最小化を図る手法である．

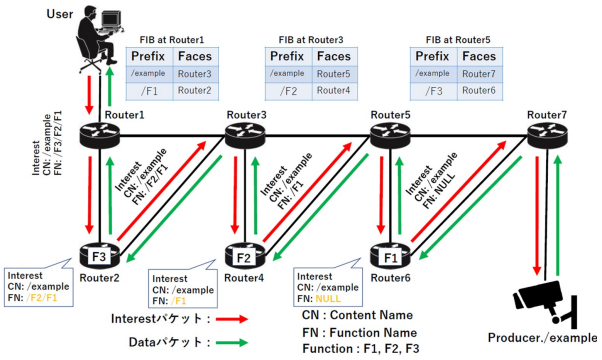

図 4　SFC におけるフォワーディングの流れ

3.4 NDN におけるキャッシュと SFC

本研究では，[5], [4] にて提案されている NDN-FC とファンクション選択手法を採用しているが，この 2 つの研究の中ではネットワーク内の各ルータでキャッシュを行っていない．そこで本研究では更なる SFC の効率化を図るために NDN 内の各ルータにキャッシュ機能を持たせキャッシュを行う手法を取っている．

ここではなぜキャッシュ機能を持たせることが SFC の効率化に繋がると考えられるかを説明する．まず，3.2 で述べたように NDN ではルータで Data パケットをキャッシュすることができる．そしてユーザがコンテンツを要求すると，ルータにユーザが要求するコンテンツ名を含む Data パケットがキャッシュされている場合は，ルータからユーザにその Data パケットを送ることができる．これをキャッシュヒットという．ルータでキャッシュヒットすることによりサーバから送信するよりもより短時間で効率よく欲しいデータを取得できる．

キャッシュ機能は SFC を使用する場合においても有効である．キャッシュヒットが起こらないときの SFC による 2 つのファンクションの処理の流れを図 5 に，ルータ A ですでにファンクションで処理されたデータがキャッシュされており，キャッ

シュヒットが起きたときの処理の流れを図 6 に示す．図からもわかるように，SFC によって適切なファンクションで処理された Data パケットがすでにルータにキャッシュされている場合，キャッシュヒットが起きることによりファンクションの処理時間やネットワーク内のパケット転送のホップ数をみると，より短い時間，経路で SFC が実現されることになる．

従って，SFC によって適切な順序で選択されるファンクションの実行を少ない処理回数で，高速に実行完了するにあたって，ユーザとサーバの経路上にあるルータでキャッシュヒットが起きることが望ましいと考えることができる．

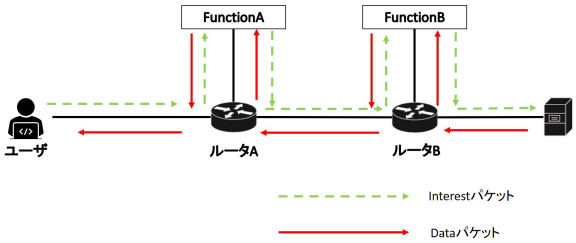

図 5　キャッシュヒットしない場合のフォワーディングプロセス

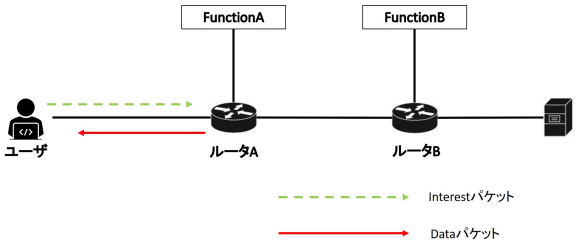

図 6　キャッシュヒットする場合のフォワーディングプロセス

4. 提 案 手 法

4.1 概　　要

本研究ではネットワーク内の各ルータにキャッシュ機能を持たせた評価を行うにあたって Caching Everything-everywhere (CEE) というキャッシュ判断方式を用いた．この方式は Data パケットが各ルータに到着したときには必ずそのパケットをキャッシュする．CEE の特性上，ユーザとサーバとの経路上に同じパケットが必ずキャッシュされてしまう．多くのパケットが転送されるネットワークにおいては各ルータにもキャッシュしておける容量は決まっているため，キャッシュリプレイスメントが頻繁に起きてしまう特徴がある．キャッシュには SFC によって選択されたファンクションによって処理されたデータと，未処理のデータが存在する．そこでキャッシュリプレイスメントが頻繁に発生してしまうと，折角 SFC によって順序付けられ適切に実行されたファンクションのデータがキャッシュとして残らなくなってしまうことがある．そうなってしまうと，キャッシュヒットしたとしてもそのヒットしたデータが未処理であって，ファンクション処理時間の削減やネットワーク内にかかる負荷の減少につながってこないという問題が発生する．そのため本研究ではキャッシュリプレイスメントが起きる際に制御を加えてファンクション処理済のデータがキャッシュヒットする確率を高める手法を提案する．

4.2 提 案 手 法

4.1 で挙げた問題点の解決策として，本研究ではキャッシュリプレイスメントの判断をする際に以下の手法を取り入れた．まず 3.1 で説明した NDN の Data パケットに，NDN-FC の Interest パケットと同様に Function Name というフィールドを追加する．変更した Data パケットを図 7 に示す．このフィールドの使い方としては，まず，Data パケットが SFC によって選択されたファンクションに到着した際に，そのファンクション名をスラッシュで区切った階層構造で Function Name に記述する．この処理を行うことによって各ルータにキャッシュされる Data パケットには，ファンクションを通過したことで Function Name にファンクション名の記載があるパケットと，ファンクションを通過していないために Function Name の記載がないパケットが存在することとなる．そして，新たな Data パケットがキャッシュされてキャッシュリプレイスメントが行われる際に，キャッシュリプレイスメントポリシーによって捨てられるパケットが 1 つ選択されるが，その選択の際に，ファンクションを通過した Data パケットを優先させ，選ばれたパケットの Function Name にファンクション名の記載があった場合は，そのパケットはリプレイスメントの対象とはせず，別のパケットを削除の対象とする．

例として，図 8 に示すようなユーザ 2 つ，サーバ 2 つ，ファンクション 4 つ，キャッシュ判断方式は CEE のときのネットワークで本提案手法を考える．このネットワークではすでにユーザ A とユーザ B がコンテンツを何度か要求しており，ネットワーク内の各ルータにはキャッシュが溜まっていることとする．

まず Function Name への記載の流れだが，ここではサーバ B から送信された Data パケットがファンクション F3 → F2 → F1 を実行してユーザ A まで転送される場合のフォワーディングプロセスを示している．Data パケットの Function Name にはファンクションを通過する際にそのファンクション名を記載するため，F3 通過後は/F3，F2 通過後は/F2/F3，F1 通過後は/F3/F2/1 となっている．

そしてキャッシュリプレイスメントを起きるときの流れだが，図では Data パケットが F3 通過後のルータ C でキャッシュリプレイスメントが起こる流れを示している．このルータ C の CS にはすでに Function Name に記載があるパケットと記載がないパケットが保持されており次にパケットをキャッシュするとキャッシュ可能な容量を超え，キャッシュリプレイスメントが起きるものとする．尚，次のキャッシュリプレイスメントによって消去されてしまうパケットはリプレイスメントポリシーにより Function Name が/F2 のパケットが選択されるものとする．ここで F3 を通過した Data パケットがルータ C に到着するとキャッシュ判断方式が CEE のため必ずルータにパケットがキャッシュされる．そこでキャッシュリプレイスメントが発生するが，ここでリプレイスメントの対象となっているパケットの Function Name には記載があるため別のパケットを

対象としてリプレイスメントを実行する．

この提案手法により，Function Name にファンクション名が追加されているパケットすなわちファンクションを通過したパケットはキャッシュリプレイスメントが起こりにくくなり，ファンクション通過後のパケットのキャッシュヒットが発生しやすくなると考えられる．

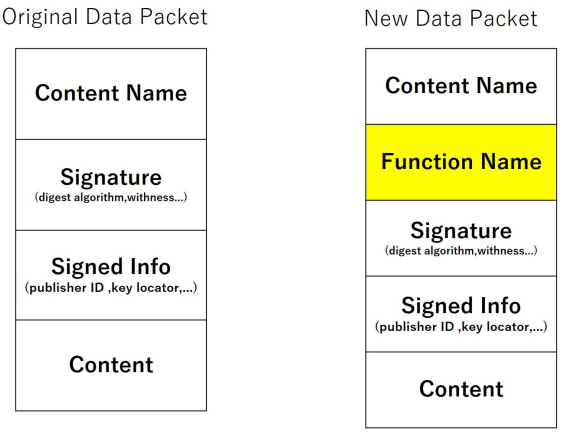

図 7　Function Name を追加した Data パケット

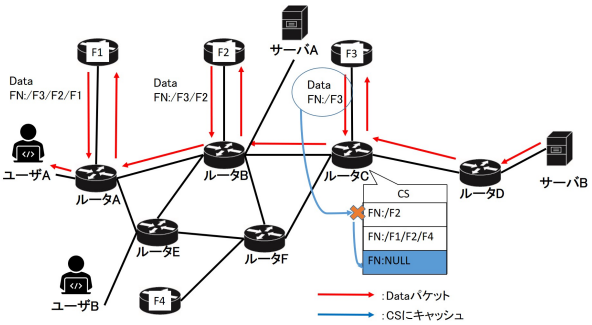

図 8　提案手法における Data パケットフローとキャッシュリプレイスメント

5.　評　　価

本研究では，SFC の実行においてキャッシュ機能を使用しない場合，キャッシュ機能は使用するがキャッシュリプレイスメントに制御を掛けない場合，キャッシュ機能を使用し，提案した手法を取り入れる場合の 3 パターンの比較をし評価を行った．

5.1　シミュレーション

評価には NDN のシミュレータである ndnSIM を用いた [1]．シミュレーションのネットワークトポロジーとしては図 9 に示す U.S- 24 トポロジーを用いた．このトポロジー上に SFC リクエストを発生させるユーザを 4 つ，サーバを 2 つ配置した．また，ユーザのコンテンツ要求は Zipf の法則にしたがって行う．評価する 3 パターン全てで共通するシミュレーション条件を表 2 に示す．さらにネットワーク上で提供されるファンクションとして，5 つの異なるファンクション F1，F2，F3，F4，F5 を用意した．それぞれのファンクションは 3 つずつのインスタンスが配備され，それぞれ a，b，c として識別する （例:F1a，F1b，F1c）．これら計 15 個のファンクションインスタンスをトポロジー上のルータに配置した．またユーザが発生させる SFC リクエストとして，表 1 に示す 12 タイプの SFC リクエストを用意した．ユーザはこの 12 タイプの中からランダムに SFC リクエストを選び，SFC リクエストを送る．この SFC リクエストに基づき，それぞれのファンクション選択手法はその手法において最も適したファンクションインスタンスを a，b，c の中から選択する．ファンクション選択手法は [4] の手法を用いており，ファンクションインスタンスの呼び出し回数と SFC を実行する際のホップ数に着目することでファンクションインスタンスの負荷を分散しつつ，サービス実行時間を最小限に抑える手法である．過去 x 回のファンクション呼び出し回数のうち，あるファンクションインスタンス j が呼び出される回数を C_j とする．このファンクション選択手法では，ある SFC リクエストを実行するファンクションインスタンスを，SFC 実行経路上の C_j の合計と SFC 実行経路のホップ数に，それぞれ重み α と β を掛けて加えた値を最小にする SFC 実行経路を選択する．本シミュレーションでは上記で説明した x の値を 30, すなわち過去 30 回のファンクション呼び出し回数のうちそれぞれのファンクションインスタンスが何度呼び出されるかを C_j とする. また重み α と β の値をそれぞれ 1 としている．また今回，キャッシュ機能を用いる 2 パターンの際は，24 個のルータ全てに同様のキャッシュ容量を持たせ，キャッシュリプレイスメントポリシーには Least Recently Used (LRU) を用いた．

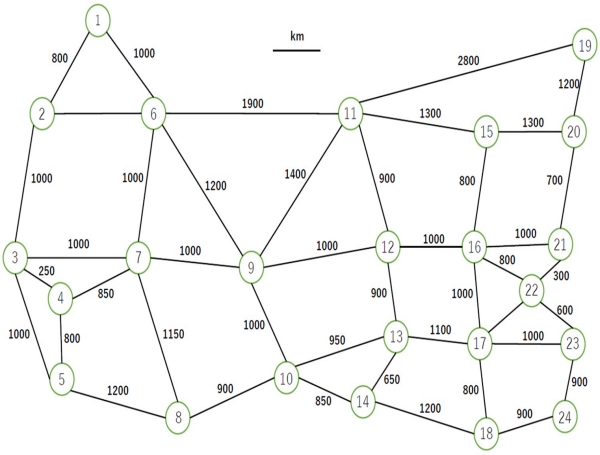

図 9　US-24 トポロジー

5.2　測 定 結 果

キャッシュ機能を使用しない場合，キャッシュ機能は使用するがキャッシュリプレイスメントに制御を掛けない場合，キャッシュ機能を使用し，提案した手法を取り入れる場合の 3 パターンのファンクション呼び出し回数の結果を以下の表 3 に示す．

5.3　考　　察

シミュレーションの結果から，キャッシュ機能を利用しない場合とキャッシュ機能を用いる場合では大きくファンクションの呼び出し回数に差が表れることがわかった．これはルータでキャッシュヒットが行われることにより，ルータから要求したコンテンツを含む Data パケットがユーザに送られ，Interst パ

表 1　SFC リクエストパターン

Type	Request
1	F1 → F2 → F4
2	F1 → F2 → F5
3	F2 → F1 → F4
4	F2 → F1 → F5
5	F1 → F3 → F4
6	F1 → F3 → F5
7	F3 → F1 → F4
8	F3 → F1 → F5
9	F2 → F3 → F4
10	F2 → F3 → F5
11	F3 → F2 → F4
12	F3 → F2 → F5

表 2　シミュレーション条件

シミュレータ	ndnSIM
トポロジ	US-24
サーバ数	4
ユーザ数	4
ルータ数	24
コンテンツ要求	Zipf 則（α =0.7）
終了条件	Data パケットが ユーザに 300 回到着時

表 3　ファンクション呼び出し回数

	ファンクション呼び出し回数
キャッシュ機能なし	1831
キャッシュ機能あり/デフォルト	867
キャッシュ機能あり/提案手法	758

ケットがファンクションを通過する機会が減ったためだと考えられる．このことから，キャッシュ機能を用いることでファンクション処理にかかる時間を削減できることが考えられる．また，キャッシュ機能を用いる場合でも，提案手法とデフォルトの手法では提案手法の方が少ないファンクション呼び出し回数に抑えられていることがわかった．これは提案手法を取りれたことにより，ファンクションで処理されたあとのパケットのキャッシュヒットが起こりやすくなったことで Interest パケットがファンクションを通過する機会が減少したと考えられる．このことから，NDN において SFC を取り入れるとき，本提案手法を使ったキャッシュ機能を用いることでファンクションの呼び出しが減り，処理の時間を削減できるため，SFC 実行の効率化を図ることができることが考えられる．

しかし，本研究の提案手法ではキャッシュリプレイスメント対象の変更を行う際に，変更対象となるコンテンツの選別ができていないため，図 8 のように，Function Name に記載がないパケットをリプレイスメントの対象とした場合だけでなく，Function Name に記載があるパケットもリプレイスメントの対象になってしまうことがある．この点を修正することでが更にファンクション処理後のパケットのキャッシュヒット率が向上すると考えらるため今後の課題として取り組んでいきたい．

6. むすび

本研究では NDN のプロトコルにおいて，SFC を実現する際にネットワーク内のルータに提案手法を用いたキャッシュ機能を持たせることで，効率の良い SFC を実行できることをファンクション呼び出し回数という指標から評価を行った．キャッシュヒットにより呼び出し回数が減少するという結果から，ファンクション処理でかかるネットワークへの負荷の軽減，要求したコンテンツ取得の短縮につながることが考えられる．

しかし本研究では，コンテンツ要求をしてから SFC によって選択したファンクションを実行し完了するまでの時間を測定していないため，今回の手法でどの程度 SFC の実行が短縮されているか正確にはわからない．今後は時間の測定も行い，より定量的にキャッシュによる SFC の効率化を示したいと考えている．また，今回の手法とは別にキャッシュヒット率をより向上させられるような手法を考えることも効率的な SFC の実現にむけての課題として挙げられる．

謝辞　本研究の成果は，総務省の（平成 30 年度）戦略的情報通信研究開発推進事業（国際標準獲得型）【JPJ000595】「スマートシティアプリケーションに拡張性と相互運用性をもたらす仮想 IoT-クラウド連携基盤の研究開発 (Fed4IoT)」によるものである．

文　献

[1] ndnsim documentation. [Online]. Available: `http://ndnsim.net/current`, Last accessed on Dec. 13, 2020.

[2] Joel Haleprn, Carlos Pignataro, and et al. Service function-chaining (sfc) architecture. In *In RFC 7665*, 2015.

[3] Named Data Networking Project. Named data networking project website. [Online]. Available: `https://named-data.net`, Last accessed on Dec. 13, 2020.

[4] Y. Shiraiwa and H. Nakazato. Function selection algorithm for service function chaining in ndn. In *2019 IEEE ComSoc International Communications Quality and Reliability Workshop (CQR)*, pp. 1–5, April 2019.

[5] 吉井 宏希, 白岩 善昭, 中里秀則. NDN における IoT データのファンクション・チェイニング. 2017 年電子情報通信学会通信ソサイエティ大会講演論文集, pp. B–8–1–9, 9 月 2017.

[6] 吉井宏希, 中里秀則. NDN におけるファンクション・チェイニングの実装. 技術研究報告 CS2018-39, 電子情報通信学会, pp. 118(128):133–138, 7 月 2018.