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
CS2020-55(2020-11)

[依頼講演]サービスファンクションチェイニング オーケストレーションとその実機評価

金井 謙治 [1)]　関根 響 [2)]　中里 秀則 [2)]　金光 永煥 [1)3)]　甲藤 二郎 [1)2)]

1) 早稲田理工学術院総合研究所 〒169-8555 東京都新宿区大久保 3-4-1
2) 早稲田大学基幹理工学部 〒169-8555 東京都新宿区大久保 3-4-1
3) 東京工科大学 〒192-0982 東京都八王子市片倉町 1404-1

E-mail: k.kanai@aoni.waseda.jp, nakazato@waseda.jp

あらまし　本稿では，計算資源および通信資源の利用率向上を可能とするサービスファンクションチェイニングオーケストレーション技術を提案する．本技術において，IoT サービスをサービスファンクションチェイニングに基づき表現し，サービスファンクションチェイニングコントローラ（ThingVisor Factory コントローラ）において，要求サービスファンクションを効率的に共有することで冗長なサービスファンクションのデプロイを抑制する．早大内に Kubernetes によるエッジコンピューティング環境を，クラウド環境にサービスファンクションチェイニングコントローラをそれぞれ構築し，本技術の実機評価を行う．

キーワード　IoT，サービスファンクションチェイニング，仮想化技術，オーケストレーション

Service Function Chaining Orchestration and its evaluations in real environment

Kenji KANAI [†]　Hibiki SEKINE[2)]　Hidenori NAKAZATO[2)]　Hirohide KANEMITSU[1)3)]
and Jiro KATTO[1)2)]

1) Waseda Research Institute for Science and Engineering (WISE), 3-4-1 Okubo, Shinjuku-ku, Tokyo, 169-0072, Japan
2) Waseda Univeristy, 3-4-1 Okubo, Shinjuku-ku, Tokyo, 169-0072, Japan
3) Tokyo University of Technology, 1404-1 Katakura-machi, Hachioji, Tokyo, 192-0982, Japan
E-mail: k.kanai@aoni.waseda.jp, nakazato@waseda.jp

Abstract　In this paper, we propose Service Function Chaining Orchestration which enables to improve utilization of computing and network resources. To realize this, IoT service is represented by a service function chaining manner and Service Function Chaining Controller (also known as ThingVisor Factory Controller) will suppress deployment and instantiation of service functions by sharing (or aggregation) of IoT service function chaining requests. We construct edge computing environment by using Kubernetes in Waseda University, deploy Service Function Chaining Controller in the cloud and evaluate performance of the proposal in the environment.

Keywords　IoT，Service Function Chaining, Virtualization technologies, Orchestration

1. はじめに

IoT サービスの普及や促進を支えるためには，より簡易に IoT サービスの構築やデプロイができるような IoT プラットフォーム技術が重要である．筆者らはこれまで 既存の IoT プラットフォーム技術や IoT プロトコル技術を活用したサービスファンクションチェイニングによる IoT サービスの構築を検討している[1, 2, 3]．特に，[2]では，サービスファンクションチェイニングによる IoT サービス構築を可能とするプラットフォーム「ThingVisor Factory」の提案，[3]では，Pub/Sub 型メッセージングモデルを活用した Pub/Sub 型サービスファンクションチェイニングの提案をそれぞれ行ってきた．

本稿では，ThingVisor Factory の要素技術の一つである IoT サービスの自動配備技術「サービスファンクションチェイニングオーケストレーション」の紹介とその実機評価結果を報告する．提案するサービスファンクションチェイニングオーケストレーションは，ユーザが設計した IoT サービスをサービスファンクションとしてネットワーク内に自動配備する技術で，各ユー

ザが要求する IoT サービスの一部，すなわちサービスファンクションを効率的にネットワーク内で共有することで，IoT サービス品質を維持したまま計算資源および通信資源の削減を可能にする．

2. 関連研究

2.1. サービスファンクションチェイニング

これまで非常に多くのサービスファンクションチェイニングの研究はなされており，ここではその関連技術についてまとめる．

サービスファンクションチェイニング（Service Function Chaining（SFC））は，ネットワーク仮想化技術（Software Defined Network（SDN））の一つとされ，類似技術として，ネットワーク機能仮想化技術（Network Function Virtualization（NFV））が挙げられる．SFC や NFV は，ネットワークのサーバ上にソフトウェアで(ネットワーク)サービスを実現する技術で，SFC は，この仮想化された NFV をサーバ内，サーバ間でネットワークを介して連結していきサービスを実現する技術である．近年では，[4]で解説しているように，エッジコンピューティングとも絡め，NFV のエッジ・クラウドへの最適配備問題[5]や割当リソース量を最適化する最適割当問題[6]，最適リソーススケーリング技術[7]などが非常に盛んに行われている．このアルゴリズムとして，近年急速に発展している深層学習ベースの手法[8]も登場してきている．

SFC を実現するうえで，これら NFV をネットワーク内で結合する際のプロトコルについても重要である．現在，IP 通信を利用するものは，パケット内に NFV を識別するための Network Service Header (NSH)タグを付与することで，そのタグを持つルータへ転送する方式が検討されており，IETF RFC7665[9]にて SFC のアーキテクチャを，IETF RFC8300[10]にて NSH タグがそれぞれ標準化されている．

これに対して，IP アドレスと NSH タグの代わりに，名前を利用して SFC のルーティングを解決する情報指向型ネットワーク(ICN/CCN/NDN)[11]に基づく SFC も多く検討されている．代表的なものとして，Function-Centric Service Chaining (FCSC) [12]や Named Function Networking (NFN) [13]が挙げられる．いずれも ICN に基づく Interest パケットに SFC におけるサービスファンクションの実行順を定義し，名前ベースでルーティングを解決している．[13]の提案の延長として[14, 15]にてプロトタイプ実装が検討されており，また，[12, 13]の類似研究として ICN-FC [16]などが挙げられる．また，[17]にて ICN ベースの NFV のプロトタイプ実装を行っている．

これらの先行研究を踏まえ，筆者らも Pub/Sub メッセージモデルに基づく Topic ベースのサービスファンクションチェイニング[2]や ICN ベースのサービスファンクションチェイニング[18]を提案している．特に，[2]では，本来 ICN のリクエストレスポンス型ではなく，IP ベースの Push 型をベースに Pub/Sub のトピックを活用することで，ルーティングを解決し，また，効率的にサービスファンクションが生み出すデータやサービスファンクションそのものの共有を実現している．[18]では，ICN-FC をベースに，ICN のオープンソース実装の一つである Named Data Networking[19]が提供するライブラリ群である ndn-cxx, Named Forward Demon をカスタマイズし，そのプロトタイプ実装を行っている．さらに，[12, 13, 18]による CCN/NDN によるサービスファンクションチェイニングに基づき，さらに CCN/NDN キャッシュを有効活用することでさらに効率的なサービスファンクションチェイングを実現する非同期型 ICN サービスファンクションチェイニングを筆者らは[20]にて提案している．[20]では，NICT が開発している ICN のソフトウェア実装の一つである Cefore[21, 22]および Python ライブラリである Cefpyco[22]を利用して非同期型 ICN サービスファンクションチェイニングを実現している．本稿では，これらサービスファンクションチェイニングをネットワーク内に自動配備するサービスファンクションチェイニングオーケストレーションについて提案する．

3. ThingVisor Factroy とサービスファンクションチェイニングオーケストレーション

3.1. ThingVisor Factory

図 1 に ThingVisor Factory の主要な機能を示す．図に示すように，ThingVisor Factory は，主要な機能として，Service Designer, Service Image Factory, ThingVisor Factory Controller, Service Chain Manager, Service Deploy Manager の機能を持つ．Service Designer は，Node-RED[23]の拡張版の位置づけで，GUI を持つアプリケーション開発者とのインターフェース機能を提供する．アプリケーション開発者は，Service Designer の GUI を通して，自身のアプリケーションを開発し，Service Designer は，開発されたアプリケーションに従い，JSON 形式にてサービスファンクションチェインの定義ファイルを生成する．この時，提案する Service Designer では，Node-RED と異なりサービスファンクション間を繋げるネットワークプロトコル（例：Pub/Sub, ICN）を指定することができる．次に，Service Image Factory は，開発者によって定義されたサービスファンクションチェインに従い，必要に応じてサービスファンクションの Docker Image を生成し，レポジトリ（例：Docker Hub）へ登録を行う．この際，Service

Designer を通して，アプリケーション開発者自身でサービスファンクションの実装も可能とする．次に，ThingVisor Factory Controller は，ThingVisor の機能を実行する API を提供する．また，前述したように，ThingVisor Factory は，ThingVisor を要求に合わせて動的に作成するプラットフォームとなり，Fed4IoT の提案する仮想 IoT システム（VirIoT）[24]との連携も意識している．そのため，VirIoT の Master Controller とは，ThingVisor Factory Controller を介して VirIoT 側の API を呼び出すものとする．次に Service Deploy Manager は，必要となるサービスファンクションやサービスファンクションの候補先の通信資源，計算資源の情報に従い，サービスファンクションのネットワーク内の（ICN）ノードへの配備先を決定する． Service Chain Manager は Service Deploy Manager によって導出されたサービスファンクションの配置位置に従い，それぞれサービス間の初期経路設定やネットワーク変動に応じた動的な経路制御機能を提供する．

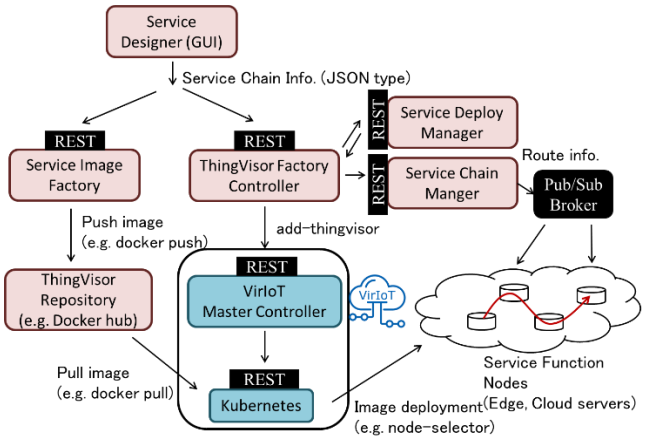


図 1. ThingVisor Factory の主な機能構成図．

3.2. サービスファンクションチェイニングオーケストレーション

サービスファンクションチェイニングオーケストレーションは，ThingVisor Factory のコンポーネントの一つで， Service Deploy Manager および Service Chain Manager の機能を総称するものである．図 2 にサービスファンクションチェイニングオーケストレーションの概念図を示す．図に示すように，ユーザからの IoT サービス要求は，サービスファンクションチェイニングとして定義され，グラフによって表現されるものとする．なお，この IoT サービス要求は，ThingVisor Factory における Service Designer により設計されるものとする．サービスファンクションチェイニングの要求を受け取った後，必要なサービスファンクションをネットワーク内に分散配備されているノードへデプロイおよびインスタンス化され，インスタンス化されたサービスファンクション間を Apache Kafka, MQTT, ICN といった通信プロトコルにより結ぶことでサービスファンクションチェイニングをネットワーク内に構築する．この時，サービスファンクションを無作為に配備すると，計算資源，通信資源の浪費に繋がるため，効率的にサービスファンクションの配備が必須となる．本技術については，2.1 節の関連研究で紹介したように NFV の最適配置問題として活発に議論・提案されている．本稿では，異なる IoT サービスの要求間におけるアグリゲーションも考慮することで，さらなる計算資源・通信資源の削減を図る．IoT サービスのアグリゲーションとして，サービスファンクションレベルのアグリゲーションを検討する．ユーザからの IoT サービス要求はサービスファンクションチェイニングとして表現されているため，IoT サービスが出力するデータは異なったとしても，その途中のサービスファンクションは一部が共通であることが多い．そこで，Service Deploy Manager では，サービスファンクションレベルで，ネットワーク内にすでにデプロイ済みであるか判定を行い，該当サービスファンクションの配備予定先に対して（通信遅延面を考慮し）地理的に近いノードにデプロイ済みであれば，重複してデプロイ（インスタンス化）をするわけではなく共有することで冗長なサービスファンクションのデプロイを防ぐ（図 3）．この機能によって，サービスファンクションのデプロイやインスタンス化にかかる通信資・計算資源を削減できると共に，サービスファンクション間を流れるデータを共有することも可能になることから，そのデータ削減にも繋がる．

ただし，現状の制約として，サービスファンクションの機能そのものの共有までは考慮しておらず，あくまで，サービスファンクションチェインの論理的な繋がりにおいて共有可能な場合のみを考慮する．つまり図 3 において，理想的なサービスファンクションの共有の場合は，SF A, SF B, SF C, SF D の 4 つをインスタンス化すれば十分であるが，例えば，SF A -> SF B -> SF C および SF A -> SF C における SF C は，要求される入力情報が SF A-> SF B と SF -> A と別物であるため，SF C は区別してインスタンス化するものとする．一方で，要求される入力情報が異なる場合においては，[20]で提案しているような非同期型サービスファンクションチェイニングを適用することで，SF C の共有は可能であることは期待される．

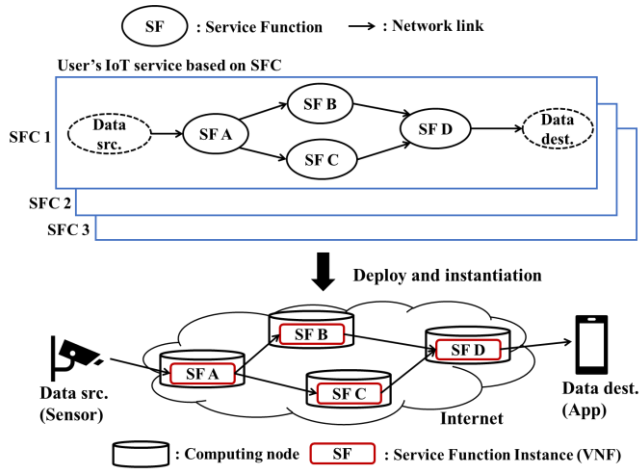
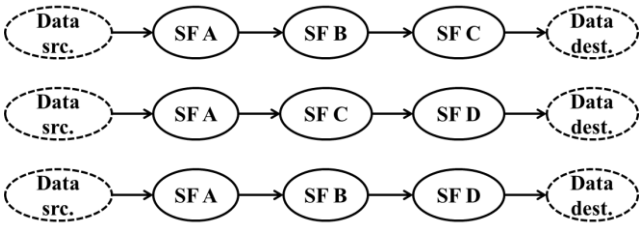
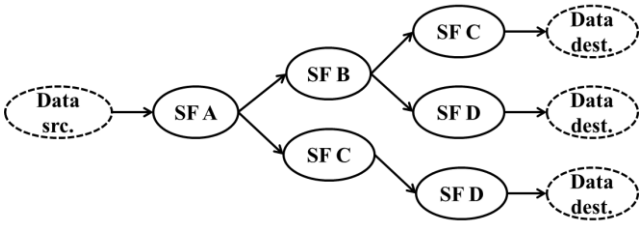


図 2. サービスファンクションチェイニングオーケストレーションの概念図.



(a) サービスファンクションチェイニングの要求



(b) アグリゲーション後の要求

図 3. サービスファンクションチェイニングのアグリゲーション例.

4. サービスファンクションチェイニングオーケストレーションの性能評価

4.1. 実験環境

実験環境を図 4 に示す. 図に示すように, エッジ環境を想定し, 早稲田大学の研究室内に Kubernetes クラスタによるエッジコンピューティング環境および Apache Kafka によるブローカーを設置した. Kubernetes クラスタとして, 4 台の Intel NUC ベアボーン PC をワーカーノード, 1 台の ZOTAC ベアボーン PC をマスターノードで構成される. ブローカーとして, コントロールプレーン用, データプレーン用の 2 台で構成される. さらに, 3 章で示す ThingVisor Factory コントローラをさくらクラウドに構築した. ThingVisor Facyory コントローラでは, 3.2 節で説明したサービスファンクションオーケストレーション機能を持つ. サービスファンクションは, 表 1 に示すように, 合計 8 種類のサービスファンクションを Docker により構築し, Docker Hub に格納しているものとする. 物体検知については, 深層学習ベースの YOLOv3-tiny[25]を利用し, GPU は利用せずに実行するものとする. 顔検知やモザイク処理は OpenCV[26]を利用し実装している. モザイク処理やカウント処理は, 物体検知あるいは顔検知で処理された結果に対して, 人物・車・動物の検知結果に対して処理するものとする. 各処理時間については, 本実験で計測された値の平均値を示すものとする. これらサービスファンクションのチェイニングプロトコルとして, [3]で提案している Apache Kafka による Pub/Sub 型を採用している.

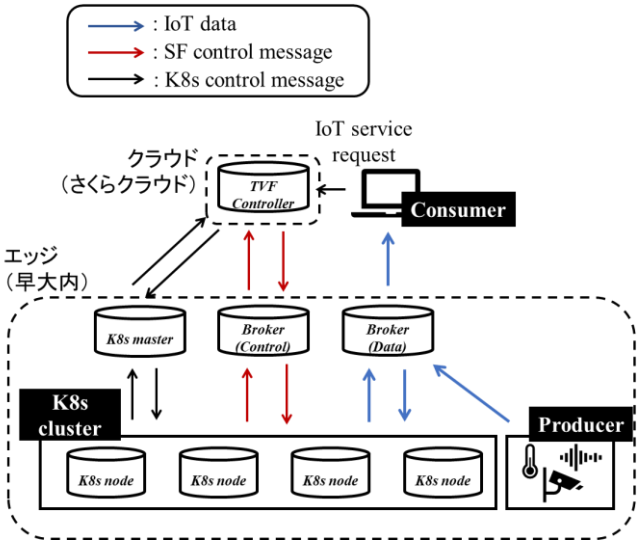


図 4. 実験環境

表 1: 実装したサービスファンクションの一覧

サービスファンクション名	必要リソース	平均処理時間
物体検知	高	1.38s
顔検知	中	0.14s
人物モザイク	小	0.02s
車モザイク		
動物モザイク		
人物カウント	小	0.003s
車カウント		
動物カウント		

4.2. 評価シナリオ

評価シナリオとして, 表 2 に実験パラメータを示す. サービスファンクションからランダムで 20 個の IoT サービス要求を生成する. 各 IoT サービス要求は 3 つ

のサービスファンクションチェインで構成されるものとする．そのため，サービスファンクションを共有せずに全てデプロイすると 60（＝20×3）のサービスファンクションが図 4 のワーカーノードにインスタンス化されることとなる．サービス要求は，10 秒ごとに要求が ThingVisor Factory コントローラへ送信されることとする．なお，IoT データ提供者（Producer）は，COCO Dataset[27]を利用し，ランダムで画像を選択し 2 秒ごとにブローカーへ送信するものとする．

サービスファンクション配備アルゴリズムとして，Kubernetes のデフォルトのスケジューラとワーカーノードをランダムで選択するランダム手法の 2 種類を利用した．それぞれで，サービスファンクションの共有（アグリゲーション）を適用する場合としない場合の合計 4 種類で評価を行う．評価メトリックとして，各サービスファンクションが生成したデータトラフィックの合計値および各ワーカーノードの合計 CPU 利用率を採用した．

4.3. 評価結果

図 5 にデータトラフィックの合計値の比較結果を図 6 に合計 CPU 利用率を示す．まず，図 5 を見ると，配備アルゴリズムによらず，サービスファンクションの共有を適用した場合において，冗長なサービスファンクションのインスタンス化を防ぐことができ，データトラフィックの大幅な削減を確認した．また，同様に，図 6 に示すように，冗長なサービスファンクションのインスタンス化に必要となる計算資源も抑えられることから大幅な計算資源の削減を確認した．

以上のことから，サービスファンクションの共有による効率的な通信資源・計算資源の利用が達成できることを確認した．

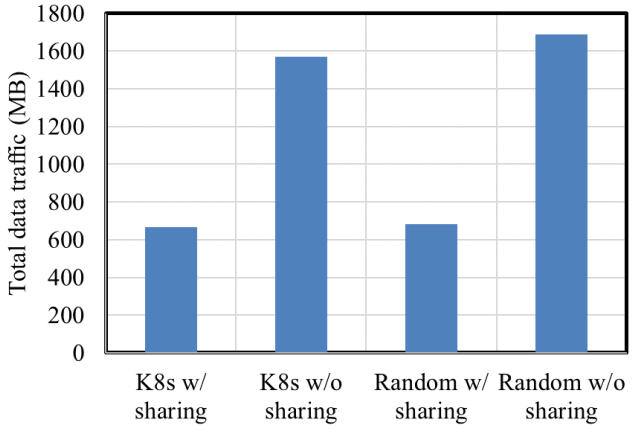


図 5. サービスファンクションが生成する合計データトラフィックの比較結果．

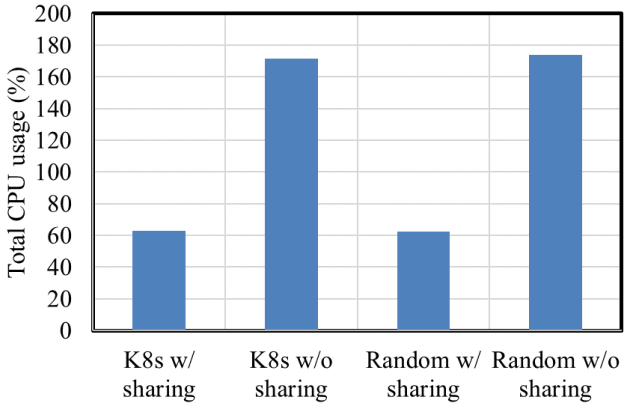


図 6. 各計算ノードにおける合計 CPU 利用率の比較結果．

5. まとめと今後の予定

本稿では，IoT サービスのネットワーク内処理において，通信資源・計算資源の利用効率向上のためのサービスファンクションオーケストレーションについて紹介した．サービスファンクションチェイニングオーケストレーションは，ThingVisor Factory の構成要素の一つに位置付けられる．本技術は，サービスファンクションチェイングとして表現された IoT サービス要求を効率的にネットワーク内に展開することを可能にするものである．配備アルゴリズムとは独立し，IoT サービス要求時に異なる IoT サービス要求間において，部分的に共通化しているサービスファンクションをネットワーク内でアグリゲーション（＝共有）することで通信資源・計算資源を効率的に削減できるものである．本技術の性能評価として，早大内に Kubernetes を利用しエッジコンピューティング環境を構築し，最大 20 サービス要求に対して，実機評価を行った．評価結果より，本技術は大幅に計算資源・通信資源を削減できることを確認した．

今後は，異なるネットワークをまたがる複数のエッジコンピューティング環境において，評価すると共に，サービスファンクションのバリエーションを増やすと共に，より多くの IoT サービス要求に対して評価していく予定である．

謝　辞

本研究成果は，戦略的情報通信研究開発推進事業（国際標準獲得型)「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)」および総務省委託研究 研究課題 VI「IoT 機器増大に対応した有無線最適制御型電波有効利用基盤技術の研究開発」技術課題ア「有無線ネットワーク仮想化の自動制御技術」の支援を受けている．

文　献

[1] 金井，吉田，金光，中里，横谷，向井，中村，上杉，“Things as a Service を実現する Fed4IoT プラットフォームの研究開発”，電子情報通信学会 信学技報 CS2019-69, pp.39-44, 2019 年 11 月.

[2] 金井，中里，金光，“Things as a Service を実現する ThingVisor Factory と IP/ICN 間の Things 共有アーキテクチャの提案”，電子情報通信学会 信学技報, vol. 119, no. 424, CS2019-101, pp. 19-24, 2020 年 2 月.

[3] Keigo Ogawa, Hibiki Sekine, Kenji Kanai, Kenichi Nakamura, Hidehiro Kanemitsu, Jiro Katto and Hidenori Nakazato, “Performance Evaluations of IoT Device Virtualization for Efficient Resource Utilization,” 2019 Global IoT Summit, Jun.2019.

[4] V. P. Kafle, Y. Fukushima, P. M. Julia, T. Miyazawa, and H. Harai, “Adaptive Virtual Network Slices for Diverse IoT Services,” IEEE Comm. Standards Magazine, vol.2, Issue.4, pp.33-41, Dec. 2018.

[5] Hidehiro Kanemitsu, Kenji Kanai, Jiro Katto and Hidenori Nakazato, “A Function Clustering Algorithm for Resource Utilization in Service Function Chaining,” IEEE CLOUD 2019, July 2019.

[6] P. Wang, C. Yao, Z. Zheng, et al., “Joint Task Assignment, Transmission, and Computing Resource Allocation in Multilayer Mobile Edge Computing Systems”, IEEE Internet of Things Journal, vol.6, Issue.2, pp.2872-2884, April. 2019.

[7] V. P. Kafle, Y. Fukushima, P. M. Julia, and H. Harai, “Scalable Directory Service for IoT Applications,” IEEE Comm. Standards Magazine, vol.1, Issue.3, pp.58-65, Sep. 2017.

[8] T. Hirayama, T. Miyazawa, M. Jibiki, and V. P. Kafle, “Service Function Migration Scheduling based on Encoder-Decoder Recurrent Neural Network,” IEEE NetSoft 2019, Jun. 2019.

[9] IETF RFC 7665 “Service Function Chaining (SFC) architecture,” Oct. 2015.

[10] IETF RFC 8300 “Network Service Header (NSH),” Jan. 2018.

[11] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, N. H. Briggs, and R. L. Braynard, “Networking named content,” ACM CoNEXT, Dec. 2009, pp. 1–12.

[12] M. Sifalakis, B. Kohler, C. Scherb, and C. Tschudin, “An information centric network for computing the distribution of computations,” ACM ICN 2014, pp.137-146, Sep.2014.

[13] M. Arumaithurai, J. Chen, E. Monticelli, X. Fu, and K. K. Ramakrishnan, “Exploiting icn for flexible management of software-defined networks,” ACM ICN 2014, pp. 107-116, Sep.2014.

[14] M. Arumaithurai, J. Chen, E. Maiti, X. Fu, and K. K. Ramakrishnan, “Prototype of an ICN Based Approach for Flexible Service Chaining in SDN,” IEEE INFOCOM Workshop, May. 2015.

[15] S. G. Kulkarni, M. Arumaithurai, A. Tasiopoulos, Y. Psaras, K. K. Ramakrishnan, X. Fu, and G. Pavlou, “Named Enhanced SDN Framework for Service Function Chaining of Elastic Network Functions,” IEEE INFOCOM Workshop, Apr. 2016.

[16] L. Liu, Y. Peng, M. Bahrami, S. Mnatsakanyan, G. Qu, Z. Ye, H. Guo, “ICN-FC: An Information-Centric Networking Based Framework for Efficient Functional Chaining,” IEEE ICC 2017, May 2017.

[17] L. Bracciale, P. Loreti, A. Detti, R. Paolillo, and N. B. Melazzi, “Lightweight Named Object: An ICN-Based Abstraction for IoT Device Programming and Management,” IEEE Internet of Things Journal, vol.6, no.3, Jun. 2019.

[18] 吉井，中里，“NDN におけるファンクション・チェイニングの実装”，電子情報通信学会 信学技報 CS2018-39, pp. 133-138, 2018 年 7 月.

[19] Named Data Networking Project [online]: https://named-data.net/

[20] 金井，中里，金光，“非同期型 ICN サービスファンクションチェイニングの提案とその実機検証” 信学技報, vol. 120, no. 75, CS2020-2, pp. 5-10, 2020 年 6 月

[21] H. Asaeda, A. Ooka, K. Matsuzono, R. Li, “Cefore: Software Platform Enabling Content-Centric Networking and Beyond,” IEICE Trans on Comm. Vo. E102 B, No. 9, pp.1792-1803, 2019.

[22] Cefore [online]: https://cefore.net/

[23] NodeRED [online]: https://nodered.org/

[24] A. Detti, G. Tropea, G. Rossi, J. A. Martinez, A. F. Skarmeta, H. Nakazato, “Virtual IoT Systems: Boosting IoT Innovation by Decoupling Things Providers and Application Developers”, Global IoT Summit, 2019.

[25] YOLO: Real-Time Object Detection [online]: https://pjreddie.com/darknet/yolo/

[26] OpenCV: Open Computer Vision Library [online]: https://opencv.org/

[27] COCO (Common Objects in Context) Dataset [online]: https://cocodataset.org/#home