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
CS2020-72,IE2020-31(2020-11)

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

関根　響† 　金井　謙治† 　甲藤 二郎† 　金光　永煥‡† 　中里　秀則†

† 早稲田大学 〒169-8555 東京都新宿区大久保 3-4-1
‡ 東京工科大学 〒192-0982 東京都八王子市片倉町 1404-1
E-mail: † {h_sekine,kanai,katto}@katto.comm.waseda.ac.jp

あらまし　筆者らはこれまで, ネットワーク機能仮想化とサービスファンクションチェイニングを利用した, SFC オーケストレーションを提案してきた．本稿ではこのフレームワークにおいて，効率的なリソース利用を実現するため，仮想ネットワーク機能の配置問題を適用する．配置手法として，SF インスタンスと通信の両方の数を最小化する目的関数を設定し，整数線形計画法に基づく数学モデルを構築する．これを検証するため，フレームワークのモデルを実装し，数値評価と実機実験を実行して性能を評価する．評価結果から，提案手法により SF 配置数と SF インスタンス間の通信数を削減できることが確認された．

キーワード　IoT，Network Function Virtualization (NFV)，Service Function Chaining (SFC)，VNF/SFC placement，Orchestration

Performance Evaluations of Service Function Chaining Orchestration

Hibiki Sekine† 　Kenji Kanai† 　Jiro Katto† 　Hidehiro Kanemitsu‡† 　and 　Hidenori Nakazato†

† Waseda University　3-4-1, Okubo, Shinjuku-ku, Tokyo, 169-8555, Japan
‡ Tokyo University of Technology　1404-1, Katakuramachi, Hachioji City, Tokyo, 192-0982, Japan
E-mail: † {h_sekine,kanai,katto}@katto.comm.waseda.ac.jp

Abstract　Authors have proposed SFC orchestration using network function virtualization and service function chaining. In this paper, we apply a placement problem of virtual network functions in order to use resources efficiently. As a placement method, we set an objective function as minimizing both numbers of SF instances and communications and build a mathematical model based on Integer Linear Programming. To validate this, we implement a model for the framework and evaluate the performances by carrying out a numerical evaluation and a real experiment. From the evaluation results, we confirm that the proposed approach can reduce the number of SF placements and the number of communications among SF instances.

Keywords　IoT，Network Function Virtualization (NFV)，Service Function Chaining (SFC)，VNF/SFC placement，Orchestration

1. はじめに

現在，Internet of Things (IoT)は，社会インフラ，エネルギー，教育など，様々な分野で広く普及している[1, 2]. その結果 IoT 技術は，市民が様々なサービスを受けることができるスマートシティの実現を加速化させることが期待されている．IoT アプリケーションは遅延やエネルギー消費，トラフィック量，処理負荷といった様々なサービス要求を持つので，IoT アプリケーションプロバイダやスマートシティ開発者は低コストでそのような多様性に応える必要がある.それゆえ，コストを削減し，スマートシティビジネスを長く持続させるためには，IoT アプリケーション開発を単純化し，大規模な IoT 展開をサポートすることが必要である．

この需要に対処するため，現在，多くの研究者はエッジ/フォグ/クラウドコンピューティングを含む，コンピューターやネットワークの仮想化技術の研究に取り組んでいる．仮想化技術は，様々なユーザーの要件を満たすため，計算および通信リソースの物理ハードウェアコンポーネントを抽象化することができる．特にネットワーク機能仮想化 (Network Function Virtualization (NFV))は，柔軟で効率的なネットワークおよびサービス管理の実現が期待されている[3]. NFV は，スイッチ，ルーター，ファイアウォールなどのネットワーク機能を仮想マシンやコンテナ上で Virtual Network Function (VNF)としてソフトウェア実装し[4],

ネットワークのカスタマイズとプログラマビリティを可能とする．VNF によって，ネットワークプロバイダは，ネットワークサービスを迅速に展開し，より柔軟かつ動的な管理ができる．またサービスファンクションチェイニング(SFC)を使用することで VNF を適切な順序にてネットワーク内で呼び出すことできる[5].

このような背景を踏まえ，筆者らは[6]にて，NFV を IoT 領域へ応用し，効率的なリソース利用を提供するために, NFV と SFC に基づく IoT サービス仮想化を提案した．筆者らの提案では，IoT サービスを IoT 指向のサービスファンクション(SF)として細かな処理単位に分割し, SF をエッジサーバとクラウドサーバに配置し，SFC に従ってネットワーク経由でチェインしている．さらに，SF の割り当てられたリソースをスケーリングすることにより，IoT サービス仮想化のパフォーマンスを確認した．また[7]にて，計算リソースおよび通信リソースの利用効率向上を可能とする SFC オーケストレーション技術を提案している．実機評価によって効率的な計算および通信リソースの利用が達成できることを確認した．ただし，これらの論文では，SF 配置の最適化が欠けている．

本稿では，SFC オーケストレーションにおいて，整数線形計画法(ILP)を適用して VNF/SF 配置問題を数学的に定式化する．この定式化では，計算リソースと通信リソースの両方の使用を最適化することを目的としているため，目的関数は SF インスタンス数と SF インスタンス間の通信回数の最小化として定義される．パフォーマンスを検証するために，Docker[13]と Kubernetes[14]を使用してフレームワークを実装し，数値計算と実機実験により性能評価を行う．

2. 関連研究

現在，多くの研究が NFV と SFC に焦点を当てている. [8]によると, NFV リソース割り当て問題は, (i)VNF チェイン構成, (ii)VNF フォワーディンググラフエンベッディング(VNF-FGE)，(iii)VNF スケジューリング(VNF-SCH)の 3 つのタイプの問題に分類される．本稿では VNF 配置問題とも呼ばれる VNF-FGE に注目する. [9]によると, 配置問題はサービス品質(QoS), エネルギー，コスト，リソース使用量，信頼性，および負荷分散の 6 つの目的に分類される.

Moens ら[10]は，実行中のサーバー数を最小限に抑えるために，ILP モデルを適用することによって最適化問題を考案し，小規模なネットワークシナリオで提案モデルのパフォーマンスを評価した. 同様に Luizelli ら[11]は，VNF インスタンス数を最小限に抑えるために，ILP モデルを提案し，ヒューリスティックなアプローチを採用して，多数のインスタンスに対応している．数値評価から，提案モデルはエンドツーエンド遅延を最大 25%削減し，許容可能なリソースのオーバープロビジョニングを 4%に制限することに成功した．さらに Ren ら[12]は，SFC を IoT 領域に拡張するために，SFC オーケストレーションスキームを検討し，混合 ILP モデルを適用することで VNF 配置戦略を提案した. 彼らのアプローチでは目的関数は SFC のホップ数の最小化である.

これらの研究を受けて, 我々は VNF インスタンス数と通信数を最小限に抑えるために ILP モデルに基づいて VNF 配置手法を提案する．さらに，Docker と Kubernetes を使用して SFC オーケストレーションを実装し, 数値評価と実機環境を実行して VNF 配置手法の性能を評価する.

3. サービスファンクション配置戦略

[7]で述べているように, SFC オーケストレーションにおいて, SF 配置戦略は効率的なリソース使用率を実現するために重要な役割を果たす．第 2 章で述べたように多くの研究者が SF 配置方法を提案しているが，SF インスタンス数と通信数を最小限に抑えるように SF 配置モデルを定式化する. SFC オーケストレーションのプロトタイプ実装を検討し，実機環境での性能を評価するため，ILP 問題に基づく単純な配置モデルを採用する.

3.1. サービスファンクション配置問題の概要

ここで SF 配置問題は，要件と制約を満たすための最適な SF 配置計画を決定することであり，インスタンス化される SF と，SF インスタンスが割り当てられる場所を決定する. 各 IoT サービス(または IoT-SFC)には，制約条件としてエンドツーエンド遅延の上限が割り当てられているため，モデルはこの要件に従って問題を解く必要がある. SF インスタンスと通信の両方の数を最小限に抑えるために, IoT-SFC が同じ SF を要求する場合, SF インスタンスを異なる IoT-SFC 間で共有できると考える．モデルパラメータは表 1 にまとめている.

3.2. ネットワークトポロジーモデル

ネットワークトポロジーモデルは，コンピューティングノードとネットワークリンクを構成し，有向グラフ G=(N,L)として表される．ここで N はコンピューティングノードのセット，L はリンクのセットである．各コンピューティングノードには CPU リソース C とその処理速度(クロック周波数)S がある. 各ネットワークリンク L はノード m と n を接続する単方向リンクを表し, そのリンク速度(帯域幅)は$bw_{m,n}$と表される. L と N はそれぞれ$(m,n) \in L$と$(m,n \in N)$を満たさなければならないことに注意が必要である.

3.3. SFC モデル

SF のセットは$F = \{f_1, f_2, \dots, f_K\}$と表し，K は SF の数を表す．SFC のセットは$R = \{V_1, V_2, \dots, V_r\}$として示され，SFC は$V_r = \{v_r^1, v_r^2, \dots, v_r^{I_r}\}$のように SF の要求のセットとして示される．ここでv_r^iは SFC V_r の i 番目の SF を表し，I_rは SF 要求の数を表す．v は SF のセットから要求されるため，$v \in F$が満たされる必要がある．

さらに，SF は IoT 関連のデータ処理を実行するため，SF はタスクに応じて入力データサイズを変換する．したがって，このようなデータサイズスケールはスケーリング係数$\alpha_{v_r^i}$で表し，SF の出力データサイズ$d_{v_r^i}^{out}$は次の式で計算できる．

$$d_{v_r^i}^{out} = d_{v_r^i}^{in} \cdot \alpha_{v_r^i}$$

さらに，SF の処理時間$t_{v_r^i}$は SF のクロックサイクル数$O_{v_r^i}$とノードのクロック周波数Sから計算され，式は次のように表される．

$$t_{v_r^i} = \frac{O_{v_r^i}}{S}$$

3.4. CPU リソース割り当てモデル

図 1 に SF の CPU リソース割り当てモデルを示す．このリソース割り当てモデルは論文[11, 15]に基づいている．図に示すように，円はコンピューティングノード上の CPU コア間の合計 CPU リソースを表し，色付きの領域は SF からの CPU リソース要件を表す．リソースの割り当ては灰色の領域を色で塗りつぶすこととして表すことができる．リソースが不足しているために SF を割り当てることができない場合，SF は他のコンピューティングノードにインスタンス化される．

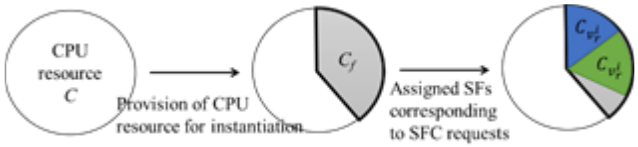

図 1. サービスファンクションへの CPU リソース割り当て

表 1. パラメータ定義

Parameters	Descriptions
N	Set of nodes
L	Set of physical links
G	Network topology graph, $G = (N, L)$
F	Set of SFs, $F = \{f_1, f_2, \dots, f_K\}$
K	Number of SFs
R	Set of SFCs, $R = \{V_1, V_2, \dots, V_r\}$
V	SFC request
v_r^i	i^{th} SF for SFC V_r
I_r	Number of SFs in SFC
C	Total CPU resources at computing node
S	Clock frequency at computing node
O_f	Number of clock cycles of SF f
C_f	CPU resource requirement for SF instance
$C_{v_r^i}$	CPU resource requirement for SF v_r^i
T_r	Upper-bound of end-to-end delay request
$bw_{m,n}$	Bandwidth between nodes m and n
$\alpha_{v_r^i}$	Scaling factor for input data size in SF v_r^i
$d_{v_r^i}^{out}$	Output data size of SF v_r^i
$d_{v_r^i}^{in}$	Input data size of SF v_r^i

3.5. 変数

ここで 3 つの決定変数を定義する．これらの変数はモデルの出力である．

・$\boldsymbol{x_f^n} \in \{\mathbf{0}, \mathbf{1}\}$：SF f のインスタンスがノード n にマッピングされているかどうか．$x_f^n = 0$の場合，SF インスタンスはマップされていない．

・$\boldsymbol{y_{v_r^i}^n} \in \{\mathbf{0}, \mathbf{1}\}$：SFC V_r の i 番目の SF がノード n にデプロイされているかどうか．$y_{v_r^i}^n = 0$の場合，SF はデプロイされていない．

・$\boldsymbol{z_{v_r^i, v_r^{i+1}}^{m,n}} \in \{\mathbf{0}, \mathbf{1}\}$：SF 同士が物理リンク$(m, n)$によって接続されているかどうか．$z_{v_r^i, v_r^{i+1}}^{m,n} = 1$の場合，SF 同士は物理リンクによって接続される．$z_{v_r^i, v_r^{i+1}}^{m,n} = 0$の場合，両方の SF は同じノードにデプロイされる．

3.6. ILP の定式化

上記を踏まえ，ILP モデルによって SF 配置問題を定式化する．

Objective function:

$$\min \sum_{n \in N, f \in F} x_f^n + \sum_{(m,n) \in L, r \in R, i \in I_r} z_{v_r^i, v_r^{i+1}}^{m,n}$$

Subject to:

$$\sum_{n \in N} y_{v_r^i}^n = 1 \quad \forall r \in R, i \in I_r \tag{1}$$

$$y_{v_r^i}^n \le x_f^n \quad \forall n \in N, r \in R, \mathrm{i} \in I_r \tag{2}$$

$$\sum_{(m,n) \in L} z_{v_r^i, v_r^{i+1}}^{m,n} \le 1, \quad \forall r \in R, i \in I_r \tag{3}$$

$$\sum_{m \in N} z_{v_r^i, v_r^{i+1}}^{m,n} - \sum_{m \in N} z_{v_r^i, v_r^{i+1}}^{n,m} = y_{v_r^{i+1}}^n - y_{v_r^i}^n, \quad \forall r \in R, i \in I_r, n \in N \tag{4}$$

$$\sum_{f \in F} C_f \cdot x_f^n \le C^n, \quad \forall n \in N \tag{5}$$

$$\sum_{r \in R} C_{v_r^i} \cdot y_{v_r^i}^n \le C_f \cdot x_f^n, \quad \forall n \in N, f \in F \tag{6}$$

$$\sum_{n\in N, i\in I_r} t_{v_r^i}^{n} \cdot y_{v_r^i}^{n} + \sum_{(m,n)\in L, i\in I_r} \frac{d_{v_r^i}^{out}}{bw_{m,n}} \cdot z_{v_r^i, v_r^{i+1}}^{m,n} \le T_r \,, \quad \forall r \in R \tag{7}$$

目的関数は，SF インスタンスと通信の両方の数を最小限に抑えることを目的としている．制約 1 は SFC の SF が 1 つのノードにのみ展開されることを保証する．制約 2 は SFC の SF がノード n で割り当てられている場合，SF に対応する少なくとも 1 つの SF インスタンスがノード n でホストされていることを保証する．制約 3 は SF が次の 1 ホップ内にあるか，同じノードで実行される次の SF と通信することを保証する．制約 4 はフローの保存を保証する．制約 5 はノード n にマップされた SF インスタンスの CPU リソースの合計が，ノード n で使用可能な物理 CPU リソースの量を超えないようにする．制約 6 はデプロイされた SF リソースの CPU リソースの合計が，SF インスタンスのマップされた CPU リソースの合計を超えないようにする．制約 7 はエンドツーエンドの遅延要求の上限が満たされる必要があることを保証する．

4. 評価実験

本章では，数値評価と実機実験を実行して，SF 配置戦略の性能を評価する．

4.1. 数値評価

まず SF 配置手法に関して数値評価を行った．評価では，最適化問題のソルバーとして Coin-or branch and cut (Cbc)[16]を使用する．表 2 は評価パラメータをまとめたものである．表に示すように，7 つのコンピューティングノードを設定し，各コンピューティングノードに CPU リソースとクロック周波数を割り当てた．またコンピューティングノード間で帯域幅を設定する．さらに SFC 要求を構成するために 5 つの SFC を生成し，各 SFC は 3~5 つの SF から構成される．各 SF には CPU リソース要件が割り当てられ，各 SFC には制約条件に対するエンドツーエンド遅延要求の上限がある．なお，設定値は各試行でランダムに選択され，30 回の試行を試みる．

表 2. 数値評価のパラメータ

Parameters	Values
Number of nodes	7
Total CPU resources of node (cores)	4 – 12
Clock frequency of node (GHz)	1 – 3
Bandwidth between nodes (Mbps)	50 – 300
Number of SFCs	5
Number of SFs in SFCs	3 – 5
CPU requirements of SF instance (cores)	2 – 4
CPU requirements of SF (cores)	0.5 – 2
Clock number of SF (clock)	1 – 3
Scaling factor for input data size	0.5 – 1.2
Initial input data size (Mbit)	1 – 20
Upper-bound of end-to-end delay request (s)	5 – 7

モデルの目的は SF インスタンス数と通信数を最小限に抑えることであるため，これらの数を評価指標として使用する．さらに，SFC のエンドツーエンド遅延を検証して，SFC がエンドツーエンドの遅延要求を超えてはならないという制約を満たすようにする．今回は比較のために，次の 4 つの典型的な SF 配置アルゴリズムを採用する．

a)“**MinIns**”: SF インスタンス数のみを考慮し，SF 配置を最適化して SF インスタンス数を最小限に抑える．

b)“**MinComm**”: 通信の数のみを考慮し，通信の数のみを最小化しようとする．

c)“**K8sLike**”: CPU リソースが最も残っているコンピューティングノードを選択し，選択したノードに SF を割り当てようとする．このアルゴリズムでは SF インスタンスは SFC 間で共有されない．Kubernetes に実装されているアルゴリズムであると大まかに考えられる．

d)“**Random**”: コンピューティングノードをランダムに選択し，選択したノードに SF インスタンスを割り当てる．

図 2 はインスタンス数と通信数の合計の平均結果を示し，図 3 はエンドツーエンド遅延の平均結果を示している．結果から，提案したアルゴリズムが目的を達成していることを確認した．

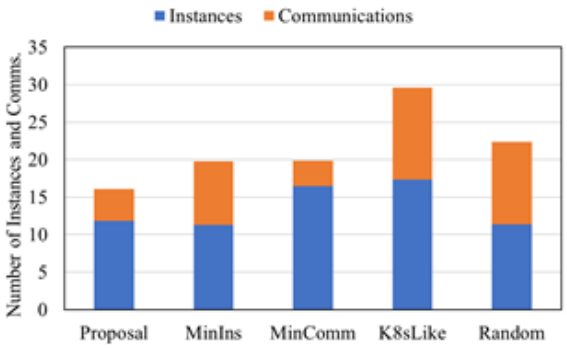

図 2. 平均インスタンス数と通信数の結果

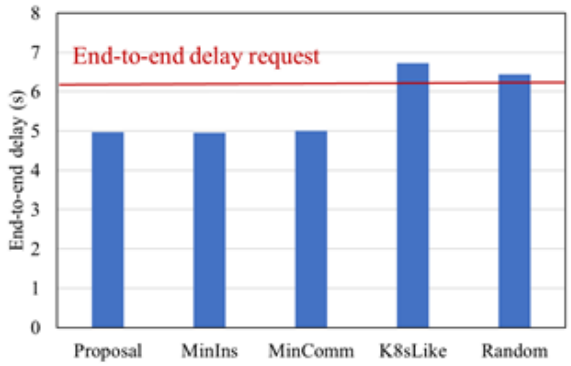

図 3. 平均エンドツーエンド遅延の結果

4.2. 実機評価

次に実機環境において性能評価を行った．図 4 に実験環境を示す．図に示すように，Kubernetes ワーカーノードのコンピューティングノードとして 5 台のデスクトップ PC を使用する．SFC の通信プロトコルとして Pub/Sub モデル(Apache Kafka[17])を採用し，5 つのノードがイーサネットケーブルを介して 100Mbps で Kafka ブローカーに接続される．ソースノードは 600×400 の画像をワーカーノードに送信し，ノードがデータを含む要求を受信すると，ノードは要求された SF を実行し，結果を次のノードに転送する．

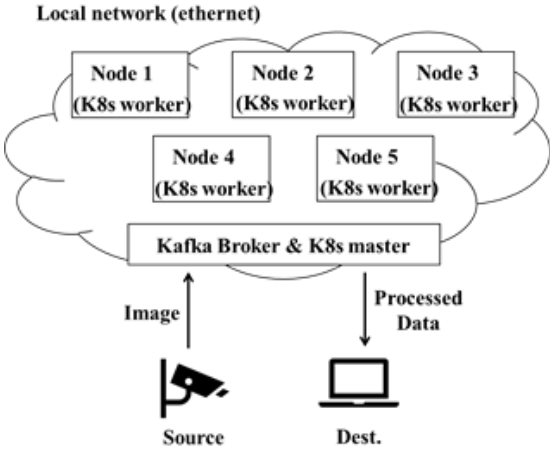

図 4. 実験環境

性能評価では，6 つの SF を実装する．YOLOv3-tiny[18]を使用した 2 つのオブジェクト検出機能(yolotiny1 と yolotiny2)，Microsoft Face[19]を使用した 2 つの顔検出機能(face1 と face2)，1 つの画像トリミング機能(crop)，1 つの画像モザイク機能(mosaic)の 6 つである．crop と mosaic の機能は OpenCV[20]によって実装されている．これらの SF は Docker イメージとして実装され，表 3 に事前に計測した各 SF の処理時間 t を示す．アプリケーションでは，V1={yolotiny1, face1, crop}, V2={yolotiny1, face2, yolotiny2}, V3={yolotiny2, face1, mosaic}の 3 つの SFC を使用する．数値評価と同様に，SF インスタンスと通信の数，およびエンドツーエンドの遅延という同じ評価メトリックを使用する．事前に，SF 配置計画は 5 つのアルゴリズムによって定義される．設定パラメータは表 4 にまとめられており，評価では 30 回の試行を実行する．

表 3. SF の処理時間

Parameters	Values
YOLOv3-tiny (sec)	1.4 – 2
Face detection (sec)	1
Crop / mosaic (sec)	0.1

表 4. 実機環境の設定パラメータ

Parameters	Values
Number of nodes	5
Total CPU resources of node (cores)	8
Bandwidth between nodes (Mbps)	100
Number of SFCs	3
Number of SFs in SFC	3
CPU requirement of SF instances	2 – 4
CPU requirement of SF	0.5 – 1.5
Initial input data size (Mbit)	1.2 – 1.6
Upper-bound of end-to-end delay request (s)	4 – 6

図 5 はインスタンス数と通信数の合計の平均結果を示し，図 6 はエンドツーエンド遅延の平均結果を示している．その結果，提案アルゴリズムでは数値評価と類似した特性であり，提案アルゴリズムは実機環境で目的を達成したことを確認した．

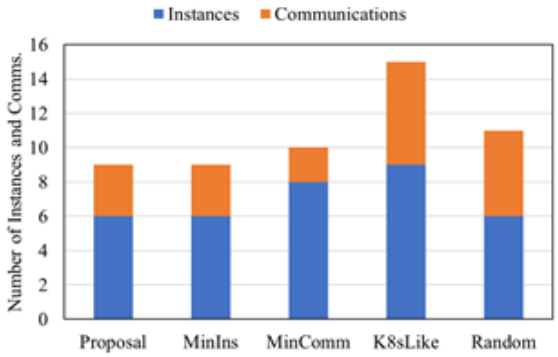

図 5. 実機環境でのインスタンス数と通信数の結果

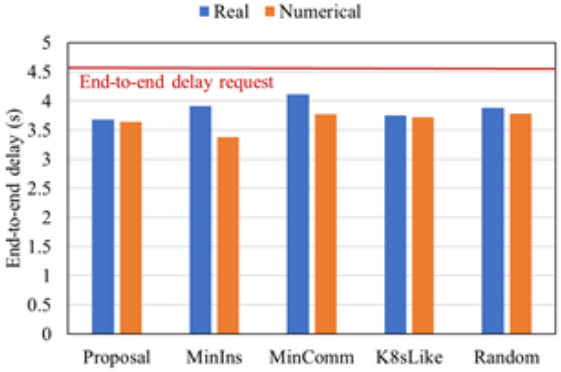

図 6. 実機環境でのエンドツーエンド遅延の結果

5. まとめ

本稿では，SFC オーケストレーションにおいて，SF インスタンス数と SF 間の通信数を最小限に抑えるべく，整数線形計画法に基づいて，サービスファンクション配置モデルを構築した．評価のために SF 配置モデルを実装し，数値評価と実機実験を実行することにより，提案モデルの性能を評価した．評価結果から，提案モデルは制約を満たすことで計算リソースと通信リソースの両方を削減できることを確認した．将来的にはディープラーニングベースのメソッドを含む他の SF 配置モデルをフレームワークに実装し，より大規模で現実的な環境を実行することによってそれらの性能を検証する．

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] J. Gubbi, R. Buyya, S. Marusic and M. Palaniswami, “Internet of Things (IoT): A vision, architectural elements, and future directions,” Future Generation Computer Systems, vol.29, no.7, pp.1645–1660, Sep.2013.

[2] A. Al-Fuqaha, M. Guizani, M. Mohammadi, et al., “Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications”, IEEE Communications Surveys & Tutorials, vol.17, no.4, pp.2347-2376, Jun.2015.

[3] R. Mijumbi, J. Serrat, J.-L. Gorricho, N. Bouten, F. D. Turck, and R. Boutaba, “Network Fucntion Virtualization: State-ot-the-Art and Research Challenges”, IEEE Communications Surveys & Tutuorials, vol.18, no.1, pp.236-262, Sep.2015.

[4] Y. Li, M. Chen, “Software-Defined Network Function Virtulization: A Survey,” IEEE Access, vol.3, pp.2542-2553, Dec.2015.

[5] A. M. Medhat, T. Taleb, A. Elmangoush, G. A. Carella, S. Covaci, and T. Magedanz, “Service Function Chaining in Next Generation Networks: State of the Art and Research Challenges”, IEEE Communications Magazine, vol.55, no.2, pp.216-223, Feb.2017.

[6] K. Ogawa, K. Kanai, K. Nakamura, H. Kanemitsu, J. Katto and H. Nakazato, “IoT Device Virtualization for Efficient Resource Utilization in Smart City IoT Platform,” IEEE PerCom 2019, Mar.2019.

[7] 金井謙治，関根響，中里秀則，金光永煥，甲藤二郎，“[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”，電子情報通信学会 CS 研究会，2020 年 11 月(発表予定)

[8] J. G. Herrera and J. F. Botero, “Resource Allocation in NFV: A Comprehensive Survey,” IEEE Trans. on Network and Service Management, vol.13, no.3, pp.518-532, Aug.2016.

[9] A. Laghrissi and T. Taleb, “A Survey on the Placement of Virtual Resources and Virtual Network Functions,” IEEE Comms. Surveys & Tutorials, vol.21, no.2, pp.1409-1434, Dec.2018.

[10] H. Moens, F. D. Turch, “VNF-P: A Model for Efficient Placement of Virtualized Network Functions”, International Conference on Network and Service Management and Workshop, Nov.2014.

[11] M. C. Luizelli, L. R. Bays, L. S. Buripl, “Piecing Together the NFV Provisioning Puzzle: Efficient Placement and Chaining of Virtual Network Functions”, IFIP/IEEE International Symposium on Integrated Network Managemetnt, May 2015.

[12] W. Ren, Y. Sun, H. Luo, M. Obaidat, “A New Scheme for IoT Service Function Chains Orchestration in SDN-IoT Network Systems”, IEEE Systems Journal, vol. 13, no.4, Dec. 2019.

[13] Docker: empowering app development for developers [online]: https://www.docker.com/

[14] Kubernetes [online]: https://kubernetes.io/

[15] Nicolas Tastevin, Mathis Obadia, Mathieu Bouet, “A Graph Approach to Placement of Service Functions Chains,” IFIP/IEEE Symposium on IM, May 2017.

[16] Coin or branch and cut (Cbc) [online]: https://projects.coin-or.org/Cbc

[17] Apache Kafka [online]. Available: https://kafka.apache.org/

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

[19] Microsoft Cognitive Services, Face [online] : https://azure.microsoft.com/en-us/services/cognitive-services/face/

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