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.

生化学反応モデルに基づくNFV管理手法のOPNFVへの実装と実験評価

杉田　修斗† 　長谷川　剛†† 　村田　正幸†

† 大阪大学 大学院情報科学研究科　〒565-0871 大阪府吹田市山田丘 1-5
†† 東北大学 電気通信研究所　〒980-8577 宮城県仙台市青葉区片平 2-1-1
E-mail: †{s-sugita,murata}@ist.osaka-u.ac.jp, ††hasegawa@riec.tohoku.ac.jp

あらまし　我々の研究グループでは、生化学反応式を用いたタプル空間モデルに基づくサービス空間構築手法の NFV 環境への適用を提案している。本報告では、提案手法を NFV フレームワーク上のコンポーネントとして実装し、実験によって性能評価を行った結果を示す。まず、オープンソースプロジェクトである Open Platform for NFV (OPNFV) を用いて NFV 環境を構築し、そこに提案手法を実現するシステムを実装した。次に、動画ストリーミングを想定したトラヒックに対して、VNF に対する CPU 資源の割り当てとサービスチェイニング要求に従った経路の設定が適切に行われ、フローのパケットを過不足なく処理できることを実験による性能評価で確認した。また、トラヒックの増減や VNF の追加等の動的な環境の変動に対しても適応的に制御が可能であることを示した。

キーワード　Network Functions Virtualization (NFV)、Virtualized Network Function (VNF)、Software Defined Networking (SDN)、Service Function Chaining (SFC)、オープンソースソフトウェア、生化学反応式

1. はじめに

Network Functions Virtualization (NFV: ネットワーク機能仮想化) システムを効率的に運用するためには、各 Virtualized Network Function (VNF: 仮想化ネットワーク機能) の配置、サーバ資源の VNF への配分、及びフローの経路等を、トラヒック量やサーバ負荷等に応じて適切に決定する必要がある。さらに、NFV システムのようなネットワークサービスは、システムの障害や需要の急激な変化等の環境変動にも素早く対応するため、自律分散的に動作することが望ましい [1]。そのような挙動を実現する方法の 1 つとして、自律分散性や自己組織性の高い、生化学機構を用いる手法がある [2]。我々の研究グループでは、生化学反応式を用いたタプル空間モデルに基づくサービス空間構築手法を、NFV に適用することを提案している [3]。この手法では、サーバをタプル空間として、サービス要求、サービス需要、及びサーバ資源等をタプル空間内の化学物質としてそれぞれ表し、サーバの挙動を生化学反応式として記述する。さらに、複数のタプル空間を接続してネットワークを構成することで、複数のサーバからなるネットワークシステムにおける、サービスや要求の移動、拡散を表現することができる。これを NFV 環境に適用することによって、分散配置された各サーバにおいて提供する VNF の決定、複数種類の VNF によるサーバ資源の共有、及び VNF の分散実行等を実現することができる。また、生化学反応式はそれぞれのタプル空間において独立に定義されて実行されるため、自律分散的な挙動を実現するのに適している。先行研究においては、コンピュータシミュレーションによる動作検証、及び簡易な実験環境での評価のみが行われている [4, 5]。

本報告では、他のコンポーネントと連携する NFV フレームワーク上のコンポーネントとして提案手法を実装することで動的 NFV 制御システムを実現し、実運用を想定した実験によってその性能評価を行う。まず、NFV を実現するソフトウェア環境の実装を目指したオープンソースプロジェクトである、Open Platform for NFV (OPNFV) [6] を用いて NFV 環境を構築し、そこに提案手法を実現するシステムを NFV フレームワーク上のコンポーネントとして設計し、実装する。次に、動画ストリーミングを想定したトラヒックを過不足なく処理できることを実験による性能評価で確認する。具体的には、`httperf` [7] を用いて生成したトラヒックについて、VNF に対する CPU 資源の割り当てと、OpenDaylight [8] アプリケーションによるサービスチェイニング要求に従った経路の設定が適切に行われた上で、フローのパケットが VNF によって全て処理され、処理遅延が発生しないことを確認する。ここで、経路制御に関しては、NFV における標準である Network Service Header (NSH) [9] を用いてサービスチェイニング要求を実現し、それに従った設定を行うことで経路制御を行う。また、トラヒックの増減や VNF の追加といった環境の動的な変動に対しても、CPU 資源の割り当てや経路の設定を適応的に行えることを示す。

2. 生化学反応モデルに基づく NFV システム

タプル空間モデル [2] は、分散システムを表現するモデルである。図 1 に、生化学反応式を用いたタプル空間モデルの概要を示す。分散システムの各コンポーネントは、タプル空間としてモデル化される。タプル空間は、生化学反応が起こる場と定義される。タプル空間内のタプルは化学物質に相当し、タプルの量は化学物質の濃度に相当する。物質濃度は、タプル空間内に化学反応式を定義することによって、増加、及び減少する。

化学反応式の実行速度は、各反応物と反応速度定数の積によって決定される。式 (1) では、反応物 X と Y の濃度がそれぞれ x と y であるとき、反応速度定数が a なので、反応速度は axy である。

$$X|Y \xrightarrow{a} Z \tag{1}$$

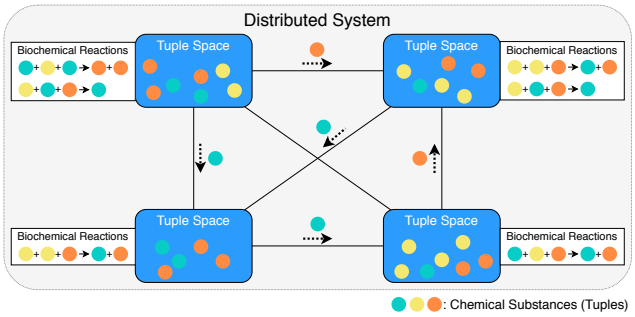


図 1　生化学反応式を用いたタプル空間モデル

この性質により、化学反応の反応速度は反応物の濃度と反応速度定数によって制御される。さらに、他のタプル空間へのタプルの拡散や移動を表現する生化学反応を定義することによって、複数のタプル空間における相互作用を表現できる。各タプル空間における化学反応は独立に発生するので、自律分散的な挙動が表現できる。

生化学反応式を用いたタプル空間モデルに基づくサービス空間構築手法を NFV へ適用したシステムで扱う反応式を式 (2)–(11) にまとめる。ここで、VNF、$RSRC$、及び PKT はそれぞれ、VNF の需要、サーバ資源、及びパケットを意味する化学物質であり、$GRAD$ はパケットの転送先を決定するための勾配場を形成するための化学物質である。また、c_0–c_{11} はそれぞれ、その反応の反応速度定数である。詳細については [5] を参照されたい。

$$\forall f \in F, VNF_f | RSRC \underset{c_1}{\overset{c_0}{\rightleftarrows}} RS\_VNF_f \tag{2}$$

$$\forall c \in C, RS\_VNF_{f_h(c)} | PKT_c \underset{c_3}{\overset{c_2}{\rightleftarrows}} MEDIATE_c \tag{3}$$

$$\forall c \in C, MEDIATE_c \xrightarrow{c_4} \begin{cases} VNF_{f_h(c)} | VNF_{f_h(c)} | RS\_VNF_{f_h(c)} \\ \quad | toserve(VNF_{f_h(c)}, PKT_c) | PKT_{c \backslash f_h(c)} & (c \backslash f \neq \emptyset) \\ VNF_{f_h(c)} | VNF_{f_h(c)} | RS\_VNF_{f_h(c)} \\ \quad | toserve(VNF_{f_h(c)}, PKT_c) & (\text{otherwise}) \end{cases} \tag{4}$$

$$\forall f \in F, VNF_f \xrightarrow{c_5} 0 \tag{5}$$

$$\forall f \in F, VNF_f \xrightarrow{c_6} VNF_f^{\rightsquigarrow} \tag{6}$$

$$\forall f \in F, VNF_f | RSRC \xrightarrow{c_7} VNF_f | RSRC | GRAD_f \tag{7}$$

$$\forall f \in F, RS\_VNF_f \xrightarrow{c_8} RS\_VNF_f | GRAD_f \tag{8}$$

$$\forall f \in F, GRAD_f \xrightarrow{c_9} 0 \tag{9}$$

$$\forall f \in F, GRAD_f \xrightarrow{c_{10}} GRAD_f^{\rightsquigarrow}(GRAD_f^-) \tag{10}$$

$$\forall c \in C, PKT_c \xrightarrow{c_{11}} PKT_c^{\rightsquigarrow}(GRAD_{f_h(c)}^+) \tag{11}$$

3. NFV フレームワーク上への実装

本章では、ETSI ISG によって提案された NFV フレームワーク [10] と、その Software Defined Networking (SDN) との統合 [11] に基づいた提案手法の実装方針について述べる。

3.1 NFV フレームワーク

図 2 に示す NFV フレームワークは、以下に示す 3 つの主要なコンポーネントから構成される。

Virtualized Network Functions (VNFs)
ネットワーク機能のソフトウェア実装であり、仮想マシン上で実行される。

NFV Infrastracture (NFVI)
VNF を実行するための基盤であり、物理資源と仮想資源の両方を管理する。物理資源は、ハイパーバイザのような仮想化レイヤを介して仮想化され、VNF を配置する仮想マシンとして提供される。

NFV Management and Orchestration (NFV MANO)
VNF や NFVI におけるライフサイクルや資源のオーケストレーションを管理する。NFV Orchestrator (NFVO)、VNF Manager (VNFM)、及び Virtualized Infrastructure Manager (VIM) という 3 つの機能が含まれる。NFVO は、Service Function Chaining (SFC) を管理し、VNF をオーケストレートする。VNFM は、VNF のライフサイクル (作成、更新、及び削除) を管理する。VIM は、NFVI におけるコンピューティング、ネットワーク、及びストレージ資源を管理し、VNF への資源割り当てを行う。

3.2 提案方式の実装方針

実運用を想定した NFV 環境への提案手法の適用のため、SDN を統合した NFV フレームワーク上に提案手法を実現する。本節では、提案手法の実装方針について述べる。

提案手法では、生化学反応式の実行結果に基づいて、VNF に対する資源割り当てとフローの経路制御を行う。フレームワーク上のコンポーネントとして提案手法を実現するために、提案手法の機能を複数のコンポーネントに分割して実装する。こうすることで、既存の NFV コンポーネントと連携させることが可能になる。実現する必要がある機能とその実装方針について、以下に述べる。

生化学反応式の計算
各サーバで定義されたタプル空間の生化学反応式の計算を行う。自律分散的な制御のため、分散実行されることが望ましい。そのため、VNF として実現するのが適切である。

フローレートと VNF 内の滞留データ量の取得
新しくシステムに入るフローに対しては、そのフローの量に相当する濃度の物質 PKT を該当するタプル空間に投入する。また、VNF 内に滞留しているパケットの量に相当する濃度の物質 PKT を該当するタプル空間の PKT の濃度とする。フローレートと滞留データ量の取得には SDN Switch からの統計情報を取得する必要があり、この機能は VIM の機能として実現するのが適切である。

VNF に対する資源割り当て
物質濃度に応じた VNF への資源割り当てを行う。この機能は VIM の機能として実現するのが適切である。

パケットフォワーディング
物質 $GRAD$ の濃度に応じたパケットフォワーディングを実現する。この機能は経路候補の管理、物質濃度に応じた経路の選択、及び SDN Switch に対するパケットフォワーディングの設定という 3 つの機能に分解することができる。1 つ目の機能は、サービスチェイニングを管理することから、NFVO の機能として実現するのが適切である。他の機能は、SDN Controller の機能として実現するのが適切である。

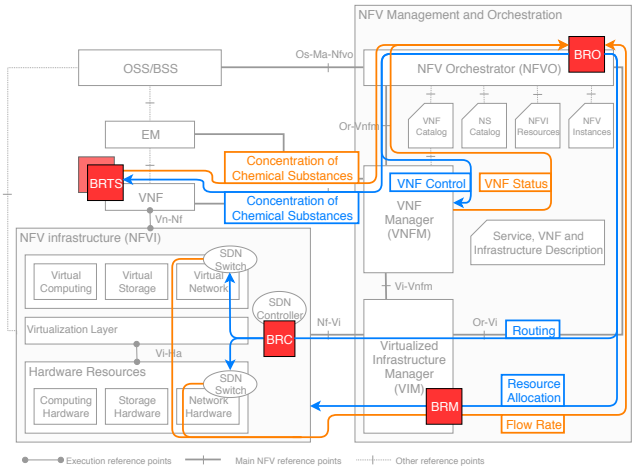


図 2　NFV フレームワーク上への提案方式の各コンポーネントの配置

コンポーネント間の物質濃度のやり取り

タプル空間が分散実行されることから、様々な機能を実現するコンポーネント間で、物質濃度のやり取りが行われる。このやり取りは、サーバの増減等の環境変動に対して適切な処理が行われるように、1 つのコンポーネントを経由して行う。この機能は NFV 全体のオーケストレーションに関わるため、NFVO の機能として実現するのが適切である。

現実の値と物質濃度の変換

物質濃度を利用するコンポーネントでは、現実の値と物質濃度の変換が必要である。コンポーネント間の物質濃度のやりとりが経由されるコンポーネントでは、各タプル空間の物質濃度を集約できることから、変換の機能もこのコンポーネントで実現する。

図 2 に、NFV フレームワーク上に提案手法を実現するためのシステムのコンポーネントの実装を示す。提案手法を実現するシステムは、以下の 4 つの主要なコンポーネントから構成される。

Biochemical Reactions Orchestrator (BRO)

提案手法を実現するシステムのオーケストレーションを行う。BRTS とは物質濃度のやり取りを行い、BRTS 間の物質のやり取りも BRO を経由する。BRC に対しては物質濃度の情報と経路の候補を提供する。BRM からフローレートと滞留データ量を取得し、BRM へ VNF ごとの資源割り当て量を伝達する。また、物質濃度と資源割り当て量やデータ量との変換を行う。NFVO のモジュールとして実現される。

Biochemical Reactions Tuple Space (BRTS)

各サーバに対応するタプル空間の生化学反応式を実行する。分散実行のためにタプル空間ごとに VNF として実現される。

Biochemical Reactions Manager (BRM)

SDN Switch からの各 VNF に関する統計情報の取得と、NFVI に対する VNF へのリソース割り当ての設定を行う。VIM のモジュールとして実現される。

Biochemical Reactions Controller (BRC)

物質濃度に基づいた確率的なパスの選択と、SDN Switch に対するパケット処理やパケットフォワーディングの設定を行う。SDN Controller のモジュールとして実現される。

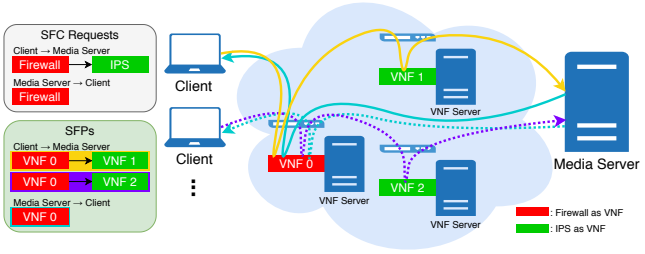


図 3　アプリケーションシナリオ

これらの具体的な実装内容については紙面の都合上省略する。

4. 実 験 環 境

4.1 アプリケーションシナリオ

本稿で行う実験では、動画ストリーミングサービスを想定したフローに対して、サービスチェイニング要求に応じた VNF の適用を行うシナリオを扱う。

想定するシナリオを図 3 に示す。この環境は 3 台の VNF サーバとクライアント群、メディアサーバから構成される。サービスチェイニング要求としては、一般的なセキュリティ対策であるファイアウォールと IPS の組み合わせを想定している。ネットワーク内には 3 台の VNF サーバが存在し、サーバ内に 1 つずつの VNF が実現されている。ファイアウォールを実現する VNF 0、IPS を実現する VNF 1 と VNF 2 が VNF として存在している。

まず、クライアントはメディアサーバに対してリクエストを送る。リクエストは NFV 環境に入り、そのフローは {Firewall → IPS} というサービスチェイニング要求を持っていると判定される。このサービスチェイニング要求に対応する SFP としては、{VNF 0 → VNF 1} と {VNF 0 → VNF 2} の 2 種類の候補が存在する。提案手法によって SFP が決定され、サービスチェイニング要求が満たされた後、メディアサーバにリクエストが到着する。

次に、メディアサーバはそのリクエストに対してレスポンスを返す。レスポンスは NFV 環境に入り、そのフローは {Firewall} というサービスチェイニング要求を持っていると判定される。このサービスチェイニング要求に対応する SFP としては、{VNF 0} の 1 種類の候補が存在する。提案手法によって SFP が決定され、サービスチェイニング要求が満たされた後、クライアントにレスポンスが到着する。

このシナリオにおけるフローの増減や VNF サーバの増減に対して、動的かつ適応的に VNF に対する資源割り当てや経路制御が行われることを確認する。

4.2 実験ネットワーク環境

本稿の実験のために構築したネットワーク環境を図 4 に示す。このネットワークはクライアント、メディアサーバ、NFV サーバの 3 台の物理サーバで構成され、それぞれは 1 Gbps のイーサネットによって接続されている。

NFV サーバでは OPNFV Fraser 6.2 を用いて NFV 環境を構築した。図 4 において、Controller0 には BRO と BRC が配置されており、また、Compute0、Compute1、及び Compute2 は VNF サーバに相当し、それぞれに BRM が配置されている。

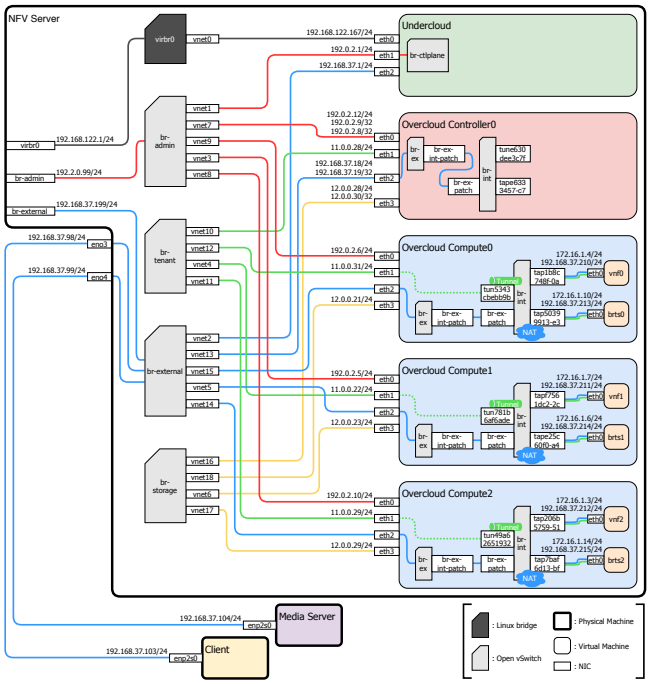
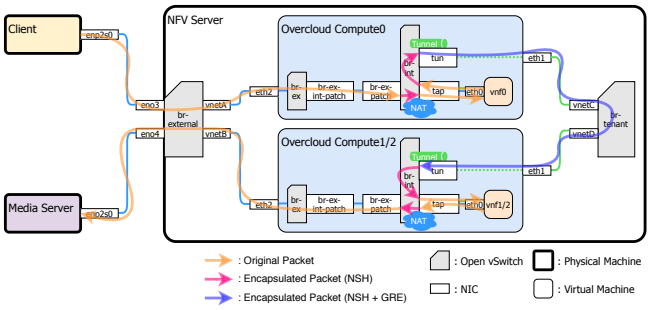
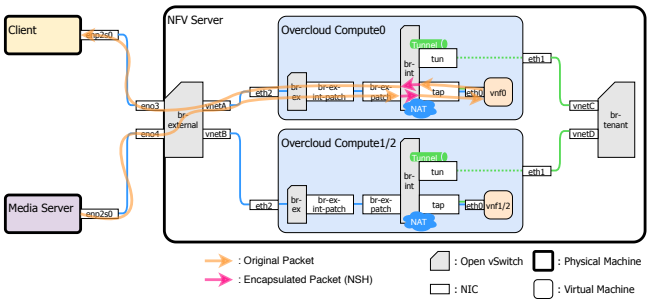


図 4　実験ネットワーク環境



(a) クライアントからメディアサーバへのフロー



(b) メディアサーバからクライアントへのフロー

図 5　パケットのルーティング

Compute0 にはファイアウォールである VNF 0 を実現する仮想マシン vnf0、Compute1 と Compute2 には IPS である VNF 1 と VNF 2 を実現する仮想マシン vnf1 と vnf2 がそれぞれ存在している。また、各コンピュートノードには BRTS を VNF として実現するための仮想マシン brts0、brts1、brts2 が存在している。brts0、brts1、及び brts2 で実現している BRTS をそれぞれ BRTS 0、BRTS 1、及び BRTS 2 と呼ぶ。BRTS 0 では Compute0、BRTS 1 では Compute1、BRTS 2 では Compute2 に対応するタプル空間が実現される。これらは OpenStack [12] のインスタンスとして実現している。メディアサーバは NGINX [13] によって実現し、動画ストリーミングサービスは HTTP によって提供される。

4.3　VNF の実現

本稿の実験で扱う VNF であるファイアウォールと IPS の実現について概説する。

ファイアウォールは Netfilter [14] によって実現し、IPS は Netfilter と Snort [15] を組み合わせて実現する。Netfilter は、Linux カーネル 2.4 以降におけるパケットフィルタリングや NAT 等のパケット処理のためのフレームワークである。Snort は、Cisco 社によって提供されているオープンソースのネットワーク侵入検知・防止システムであり、IP ネットワーク上でのリアルタイムトラヒック分析やパケットロギング等を実行することができる。Netfilter や Snort の設定においては、フィルタリングやアラートのルールの設定はせず、単純な転送のみを行うものとする。

4.4　動画ストリーミングサービスの実現

4.1 節で示したシナリオに沿った実験を行うため、動画ストリーミングサービスを想定した HTTP のフローを `httperf` [7] によって生成する。

具体的なフローの設定については、YouTube 等の動画配信サービスについて述べられている文献 [16] を参考に決定する。YouTube における 1080p の動画の再生ビットレートは約 3.60 Mbps であり、データの送信はそれより十分大きい速度で行われている。そこで採用されている 2 種類のデータ送信メカニズムの内、一方のメカニズムの後半では 32–128 KB のデータを 1 往復で取得し、平均で約 6.13 Mbps でデータを伝送している。これを模したフローを以下のように生成する。

64 KB のファイルをメディアサーバに配置し、クライアントから 1 コネクションあたりに 10 回リクエストを行うとする。この設定では、1 秒あたりに 1 コネクションを生成すると 5 Mbps のフローが流れることになり、1080p の動画の再生ビットレート以上のフローが流れることになる。

実験では、上述のコネクションを同時に複数生成することで、複数のクライアントからのフローを模する。以降では、`httperf` で設定する、このコネクションの 1 秒あたりの生成数をコネクションレートと呼ぶ。

4.5　フローの経路設定方法

OpenFlow プロトコルを用いた、4.1 節で示したシナリオに沿ったフロールーティングの具体的な実装について説明する。

本実験で扱うルーティングについて概説したものを図 5 に示す。本実験で扱う VNF は、デフォルトでは NSH に対応していないためインスタンスが接続されている仮想スイッチである `br-int` が SFC プロキシ [17] の役割を行う。フローを示す矢印の色はそれぞれパケットの状態を示しており、オレンジが元のパケット、ピンクが NSH によるカプセル化が行われたパケット、紫がその上に GRE によるカプセル化が行われたパケットを示す。なお、ヘッダの書き換えについてはこの図では表現されていない。本図に示すパケット処理を OpenFlow プロトコルを用いて実現した。詳細については紙面の都合上省略する。

5. 実験結果と考察

本章では、フローが化学物質濃度に従ってルーティングされることで負荷分散が行われ、また、フローレートに応じた CPU 資源が VNF に適切に割り当てられることを確認する。

5.1 評価シナリオとパラメータ設定

4.1 節で示したシナリオで、動的な経路制御と資源割り当ての制御が行われる場合のフローについて扱う。

`httperf` を一定のコネクションレート 10 で 100 秒間実行する。最初の 50 秒間は、IPS が実現される VNF 2 は使用できないものとし、BRTS 2 は他のタプル空間に接続されない。そのため、VNF 1 が、IPS が適用されるべき全てのフローを処理する。実験開始後 50 秒が経過した時点で VNF 2 が使用できるようになり、BRTS 2 が他のタプル空間に接続される。今回の実装では簡単のため、Compute0 ではファイアウォールのみ、Compute1 と Compute2 では IPS のみが実行されるものとする。また、BRTS 2 の他のタプル空間との接続は初めに行うものの、その段階では物質のやり取りをせず、50 秒が経過した時点で物質のやり取りを始めるように設定し、物質は BRTS 1 と BRTS 2 の間でのみ移動が行われるものとする。

BRTS 0 では、生化学反応式として式 (2)–(5) を定義し、物質濃度の初期値として、$RSRC$ は 1,000、VNF は 1,000、他の物質については 0 を設定した。BRTS 1 では、生化学反応式として式 (2)–(10) を定義し、物質濃度の初期値として、$RSRC$ は 1,000、VNF は 1,000、他の物質については 0 を設定した。BRTS 2 では、生化学反応式として式 (2)–(10) を定義し、物質濃度の初期値として、$RSRC$ は 1,000、他の物質については 0 を設定した。また、全ての反応式の反応速度定数として 0.2 を設定する。

5.2 実験結果と考察

図 6 に、実験結果として得られた、各タプル空間における化学物質濃度、システムが観測した各 VNF に対するフローレートや各 VNF 内の滞留データ量といった統計情報、各 VNF に対する CPU 資源割り当て量、クライアントからメディアサーバに向けたフローについて各 SFP が適用されたフローの数、及び `httperf` のコネクションあたりのリプライ数と実行時間の、経過時間に対する遷移を示す。

まず、IPS として VNF 1 のみが使用されている前半の 50 秒について述べる。図 6(h) より、この期間では、クライアントからメディアサーバに向けた全てのフローに対して {VNF 0 → VNF 1} という SFP が選択されていることがわかる。また、図 6(a) と図 6(b) ではフローレートに応じた値に各物質濃度が収束していることがわかる。しかし、収束までにかかる時間は約 20 秒程度である。これは全ての反応式の反応速度定数を 0.2 に設定した影響で、各生化学反応式の実行速度が小さいためである。反応速度定数の設定は、収束に至るまでの過渡特性や、反応式の計算を実行するサーバの処理性能などを考慮して設定する必要がある。具体的な設定方法については今後の課題としたい。

次に、VNF 2 が使用できるようになった後半の 50 秒間について述べる。図 6(a)–6(c) を見ると、BRTS 0 における物質濃度は前半と比べて変化はない。一方、BRTS 1 における物質濃度は、BRTS 2 が接続された直後、$GRAD$ の濃度が減少し、その後上昇することがわかる。これは、BRTS 2 に対して $GRAD$ の拡散が行われ、その後再び収束に向かうためである。また、BRTS 2 ではシステムへの接続直後から各物質濃度は収束値へ近づいていき、最終的な収束値は BRTS 1 と同様になることがわかる。

図 6(h) より、BRTS 2 の接続後しばらくすると、クライアントからメディアサーバに向けたフローに対して {VNF 0 → VNF 2} という SFP が選択されるようになり、VNF 2 が使用されるようになっていることがわかる。また、最終的には VNF 1 が使用される SFP と、VNF 2 が使用される SFP がほぼ均等な確率で選択されていることがわかる。これは各タプル空間の $GRAD$ の濃度に応じて、適切にフロールーティングが行われていることを示している。これによって各 VNF に対するフローレートも変化し、各 BRTS における物質濃度に影響を与えている。今回の実験の設定では最終的に SFP は均等に選択されたが、反応速度定数を変更することで、この制御を変更することも可能である [5]。例えばクラウドエッジコンピューティングにおいて、一定のフローレート以下ではエッジに配置した VNF のみを使用し、それ以上になればクラウドに配置された VNF も使用するように制御することも可能である。

図 6(g) を見ると、化学物質濃度が収束した後は、VNF に対する CPU 資源が正しく割り当てられていることがわかる。また、図 6(i) を見ると、全ての実験について全てのリクエストが処理され、実行時間についてもフローレートに対して CPU 資源が不足している実験開始直後を除き、増加していないことがわかる。

これらの結果から、VNF への動的な資源割当、フロー経路の動的な設定などが正しく動作していることが確認できた。

6. まとめと今後の課題

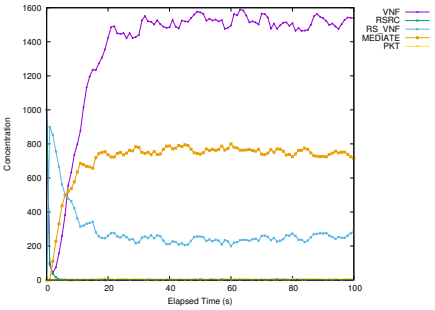
本報告では、生化学反応式を用いたタプル空間モデルに基づくサービス空間構築手法を実装し、実験によってその性能評価を行った。実装では、OPNFV を用いて構築した NFV 環境に、ETSI ISG によって提案された NFV フレームワーク上に配置した 4 つの主要なコンポーネントからなるシステムを構築し、提案手法を実現した。性能評価では、動画ストリーミングを想定したトラヒックに対して、VNF に対する CPU 資源の割り当てとサービスチェイニング要求に従った経路の設定が適切に行われ、フローのパケットを過不足なく処理できることを確認した。また、環境の動的な変動に対しても制御を適応的に行えることを示した。

今後の課題として、現実の値と物質濃度の変換に必要な VNF ごとのパラメータを機械的に決定する手法の提案や、サーバ間の伝送遅延時間やリンク帯域等の要素を提案手法へ導入することが考えられる。また、本報告で扱わなかった様々なシナリオでの実験評価も挙げられる。

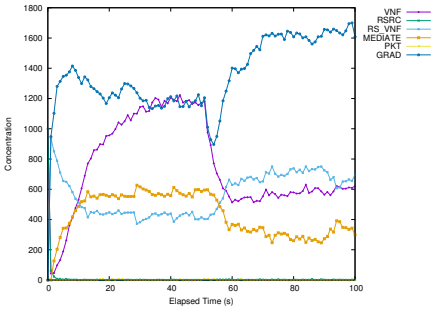
文　献

[1] I. Foster, C. Kesselman, C. Lee, B. Lindell, K. Nahrstedt, and A. Roy, “A Distributed Resource Management Architecture that Supports Advance Reservations and Co-allocation,” in *Proceedings of 1999 Seventh International Workshop on Quality of Service*, pp. 27–36, May 1999.

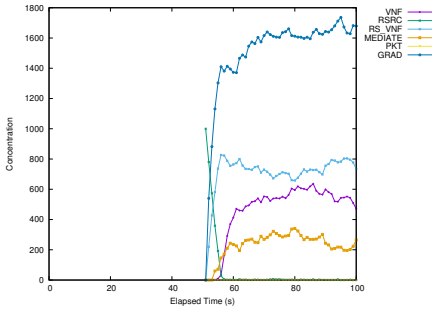
[2] M. Viroli, M. Casadei, S. Montagna, and F. Zambonelli, “Spatial Co-



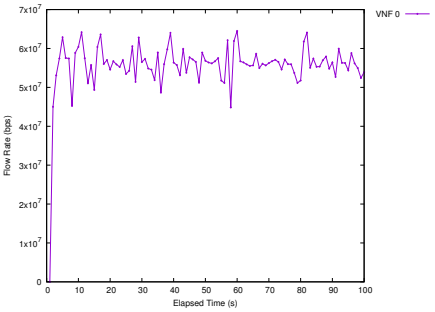
(a) BRTS 0 における化学物質濃度



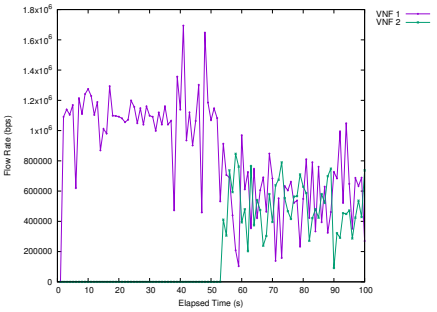
(b) BRTS 1 における化学物質濃度



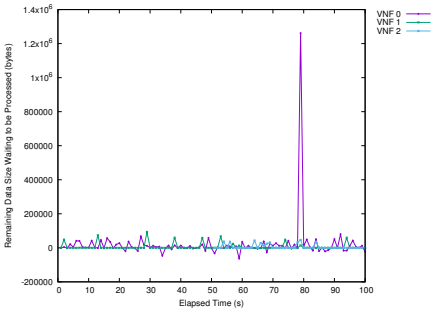
(c) BRTS 2 における化学物質濃度



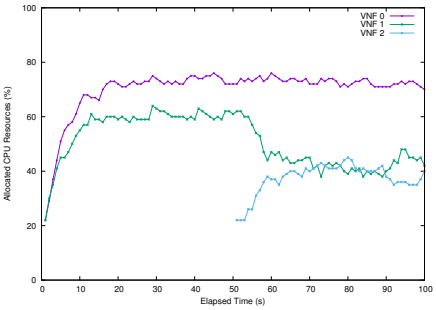
(d) VNF 0 に対するフローレート



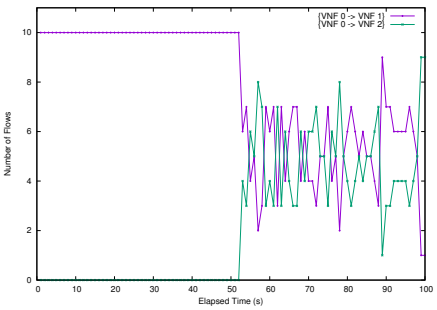
(e) VNF 1 と VNF 2 に対するフローレート



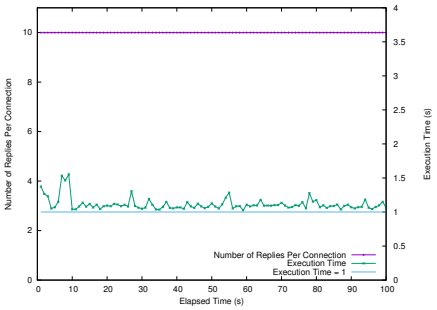
(f) 各 VNF 内の滞留データ量



(g) 各 VNF に対する CPU 資源割り当て量



(h) クライアントからメディアサーバに向けたフローについて各 SFP が適用されたフローの数



(i) `httperf` のコネクションあたりのリプライ数と実行時間

図 6　実 験 結 果

ordination of Pervasive Services through Chemical-Inspired Tuple Spaces," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 6, no. 14, pp. 1–24, June 2011.

[3] G. Hasegawa and M. Murata, "Biochemically-inspired Method for Constructing Service Space in Virtualized Network System," in *Proceedings of ICIN 2016*, Mar. 2016.

[4] K. Sakata, "Adaptive and Autonomous Placement Method of Virtualized Network Functions based on Biochemical Reactions," Master's thesis, Osaka University, Feb. 2018.

[5] R. Kurokawa, "Biochemically-inspired, Adaptive, and Autonomous VNF Control for Service Function Chaining," Master's thesis, Osaka University, Feb. 2019.

[6] "OPNFV: Home." available at `https://www.opnfv.org`.

[7] "GitHub - httperf/httperf: The httperf HTTP load generator." available at `https://github.com/httperf/httperf`.

[8] "Home - OpenDaylight." available at `https://www.opendaylight.org`.

[9] "Network Service Header (NSH)." available at `https://www.rfc-editor.org/rfc/pdfrfc/rfc8300.txt.pdf`.

[10] ETSI GS NFV 002, "Network Functions Virtualisation (NFV); Architectural Framework." available at `http://www.etsi.org/deliver/etsi_gs/nfv/001_099/002/01.01.01_60/gs_nfv002v010101p.pdf`.

[11] ETSI GS NFV 005, "Network Functions Virtualisation (NFV); Ecosystem; Report on SDN Usage in NFV Architectural Framework." available at `https://www.etsi.org/deliver/etsi_gs/NFV-EVE/001_099/005/01.01.01_60/gs_nfv-eve005v010101p.pdf`.

[12] "Home - OpenStack is open source software for creating private and public clouds." available at `https://www.openstack.org`.

[13] "NGINX — High Performance Load Balancer, Web Server, & Reverse Proxy." available at `https://www.nginx.com`.

[14] "netfilter/iptables project homepage - The netfilter.org project." available at `https://www.netfilter.org`.

[15] "Snort - Network Instruction Detection & Prevention System." available at `https://www.snort.org`.

[16] H. Hisamatsu, G. Hasegawa, and M. Murata, "Network Friendly Transmission Control for Progressive Download over TCP," *Journal of Communications*, vol. 7, no. 14, pp. 213–221, Mar. 2012.

[17] "Service Function Chaining (SFC) Architecture." available at `https://www.rfc-editor.org/rfc/pdfrfc/rfc7665.txt.pdf`.