特別研究報告

題目

生化学反応モデルに基づいた動的資源割り当て手法の NFV フレームワークにおける実験評価

指導教員
松岡 茂登 教授

報告者
杉田 修斗

平成 30 年 2 月 14 日

大阪大学 基礎工学部 情報科学科

平成 29 年度 特別研究報告

生化学反応モデルに基づいた動的資源割り当て手法の NFV フレームワークにおける実験評価

杉田 修斗

内容梗概

スマートフォン及びタブレット端末の普及や、モノのインターネット (Internet of Things) 技術の発展等により、ネットワークに接続される機器の種類や量が急激に増大している。既存機器で対応できない大量のトラヒックの収容や新しいネットワークサービスの導入を行うために、専用のハードウェア機器を増設すると、多大なコストがかかる上、環境変化に柔軟に対応できなくなる。そのような問題に対処するための技術として、ネットワーク機能仮想化技術 (Network Function Virtualization: NFV) がある。NFV においては、これまで専用のハードウェア機器によって実現されてきたネットワーク機能を仮想ネットワーク機能 (Virtual Network Function: VNF) として実現することで、複数のネットワーク機能が 1 台のサーバ資源を共有したり、1 つのネットワーク機能を複数サーバ上で分散実行したりすることが可能になる。

NFV 環境の効率的な運用のためには、各 VNF の配置、サーバ資源の VNF への配分、及びフローの経路等を、状況に応じて適切に決定する必要がある。また、このようなネットワークサービスは、環境変動に素早く対応できるように、かつ、サービス拡張性を維持するために、自律分散的に動作することが求められる。これを実現する方法の 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方法がある。

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 環境へ適用することを提案している。これにより、分散配置された各サーバで提供する 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できる。また、化学反応式はそれぞれのタプル空間で独立に定義されて動作するため、自律分散的な挙動を実現することができる。過去の研究では、コンピュータシミュレーションによる基本的な動作検証、及び簡易な実験環境での評価のみが行われている。しかし、NFV 環境への適応性を示すためには、近年実装が進んでいる NFV フレームワーク上での検証が必要である。

そこで本報告では、NFV フレームワークを用いて、上述のサービス空間構築手法を NFV 環境へ適用し、その動作を実験により確認した。具体的には、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 を用い、VNF としてネットワーク侵入検知システムの機能を持つオープンソースソフトウェアの 1 つである Snort を利用して、実験環境を構築した。次に、生化学反応モデルに基づいて、ネットワークフローのトラヒック量に応じて VNF へサーバ資源を割り当てる機構を実現し、NFV フレームワーク上での動作を確認した。実験の結果、構築したシステムが、ネットワークのトラヒック量に応じて VNF へ CPU 資源を割り当てることで、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、トラヒック量の動的な変動に対しても、割り当てる CPU 資源を適応的に調整できることを示した。

主な用語

生化学反応式、ネットワーク機能仮想化技術 (NFV)、仮想ネットワーク機能 (VNF)、NFV フレームワーク、タプル空間モデル、動的資源割り当て

目 次

図 目 次

表 目 次

1　はじめに

近年、スマートフォン及びタブレット端末が普及したことや、モノのインターネット (Internet of Things: IoT) 技術 [1] の発展等により、ネットワークに接続される機器の種類や量が急激に増大している。その結果、求められるネットワークサービスの種類が多様化し、トラヒック量も急増したことで、これまで以上のネットワーク関連設備の充実が求められている。既存の機器では対応できない大量のトラヒックや、新しいネットワークサービスの開始に対応するために、従来は専用のハードウェア機器を増設することで対応してきた。しかし、この方法では、導入機器そのものや機器を設置する物理的空間等を用意するための初期導入コストがかかる上に、機器運用のための電力コストの増加等を引き起こす。また、機器の保守及び運用や、システム障害や急な需要増加等の環境変化への対応には手間がかかり、柔軟性が低い。

このような問題に対処するための技術の 1 つとして、ネットワーク機能仮想化技術 (Network Function Virtualization: NFV) がある [2]。NFV は、専用のハードウェア機器によって実現されてきたネットワーク機能を、汎用サーバ上で動作するソフトウェアによって実現する技術である。ソフトウェアによって実現されたネットワーク機能を仮想ネットワーク機能 (Virtual Network Function: VNF) と呼ぶ。VNF として、ファイアウォール [3]、Deep Packet Inspection (DPI) [4]、ネットワークアドレス変換 (Network Address Translation: NAT) [5]、及び Evolved Packet Core (EPC) [6] 等多岐にわたるネットワーク機能が実現されている [7]。図 1 に、NFV システムを適用したネットワーク構成の例を示す。このように、ネットワーク機能を VNF として実現することによって、ネットワーク機能をソフトウェアとして管理することができる。また、複数のネットワーク機能が 1 台のサーバ資源を共有したり、1 つのネットワーク機能を複数サーバ上で分散実行することが可能になる。これによって、運用コストや設備投資コストを抑えることができ、環境変動にも柔軟に対応できる。

また、NFV におけるネットワークフローは、適用されるべき VNF の順序を示すサービスチェイニング要求を持つ。図 1 では、NFV システムにおいて、サービスチェイニング要求に従って、適当な VNF がパケットに適用される様子を示している。このような NFV システムを効率的に運用するためには、各 VNF 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 [8]。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。

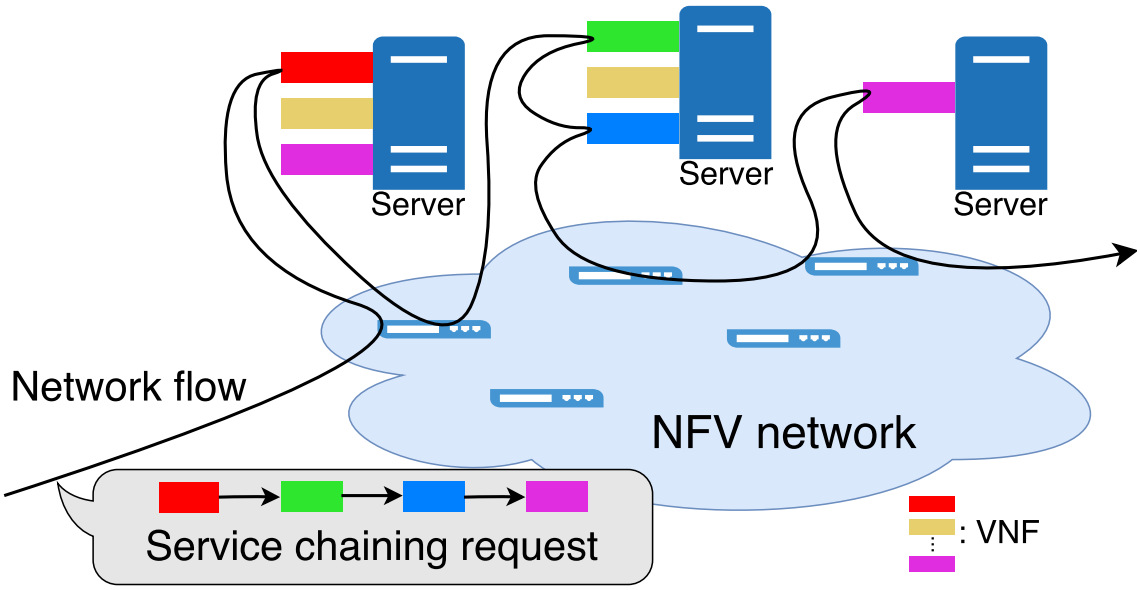


図 1: ネットワーク機能仮想化 (NFV) 環境

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 [9, 10]。この手法は、サーバをタプル空間として、サービス要求やサービス需要、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出来る。また、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、上述の手法の評価としては、コンピュータシミュレーションによる基本的な動作検証、及び簡易な実験環境での評価のみが行われている [11, 12]。しかし、NFV 環境への適応性を示すためには、近年実装が進んでいる NFV フレームワーク [13] 上での検証が必要である。

そこで本報告では、上述したサービス空間構築手法を NFV フレームワークに適用し、VNF への動的資源割り当てを実現し、NFV が実運用されるような環境での提案手法の有効性を実験で検証する。初めに、NFV 環境において必要となる、VNF のサーバへの配置、サーバ資源の配分、及びトラヒック量やサーバ負荷に応じたネットワークフローの経路決定等を実現するために、それらをサービス空間構築手法における化学反応式に対応付ける。次に、NFV 環境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) [14] を用いて NFV 環境を構築し、生化学反応モデルを適用する。VNF として、ネットワーク侵入検知システム (Instruction Detection System: IDS) 機能を持つソフトウェアの 1 つである Snort [15] を利用し、ネットワークフローのトラヒック量に応じて、サーバ上で動作する Snort のプロセスに CPU 資源が割り当てられ、フローのトラヒックを過不足なく処理できていることを確認する。

本報告の構成は以下の通りである。まず 2 章で、本報告で扱う、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説明し、NFV 環境への適用方法を示す。次に 3 章では、NFV フレームワークの概要と本報告において用いたオープンソースプロジェクトである OPNFV について説明する。4 章では、実験ネットワーク環境、及びその中で用いた VNF プログラム及びパケット送信プログラムについて説明する。また、本実験環境における、提案手法の実現方法について述べる。5 章では、本報告で行った実験の設定等を説明し、実験結果を示して提案手法の有効性を検証する。最後に 6 章で本報告のまとめと今後の課題について述べる。

2　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

本章では、文献 [12] から引用し、仮想ネットワークシステムにおける、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 [10] について述べる。また、NFV フレームワークを用いた実験環境への適用方法、及び実装方針について説明する。

2.1　化学反応式を用いたタプル空間モデル

生化学反応式を用いた 1 つのタプル空間モデルの概要を図 2 に示す。タプル空間を化学反応が起こる場とする。タプル空間内のタプルは化学物質に相当し、その量は化学物質の濃度に相当する。タプル空間内で起こる化学反応を定義することで、タプルの量の増減やタプルの挙動が定められる。タプル空間におけるそれぞれの化学反応式は、全ての反応物の濃度と、反応式に定義された反応速度係数の積に比例して実行される。また、タプル空間外からの化学物質の投入や、生成物を外部に送出する挙動を記述することによって、タプル空間外との相互作用を実現することができる。さらに、本報告では用いていないが、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、タプル空間同士のタプルの拡散や移動を表すことができる。このとき、各タプル空間で起こる化学反応はそれぞれの空間で独立に起こるため、自律分散的な動作を表現することができる。

次に、前述のモデルの仮想ネットワークシステムへの適用方法について説明する。図 2 におけるタプル空間は、サービスを提供するサーバに対応付けられる。タプル空間内のタプルは、サービス需要、サービス要求、及び資源量等を表す。サーバにおける、提供するサービスの決定、複数の種類のサービスによる資源量の共有、サービスの分散実行、及びサービスを適用するフローの決定等の挙動を、化学反応式を定義することによって表現する。以降では、1 台のサーバで運用される仮想ネットワークシステムへの適用方法について概略を述べる。

2.1.1　サービスの実行及び成長と衰退

仮想ネットワークシステムにおけるサービスは、その要求に従って実行される必要がある。また、需要の多いサービスは成長させ、需要の少ないサービスは衰退させることが求められる。このような挙動を反応式 (1) 及び (2) によって実現する。

$$SERV|REQ \xrightarrow{u} SERV|SERV|toserve(SERV, REQ) \tag{1}$$

$$SERV \xrightarrow{d} 0 \tag{2}$$

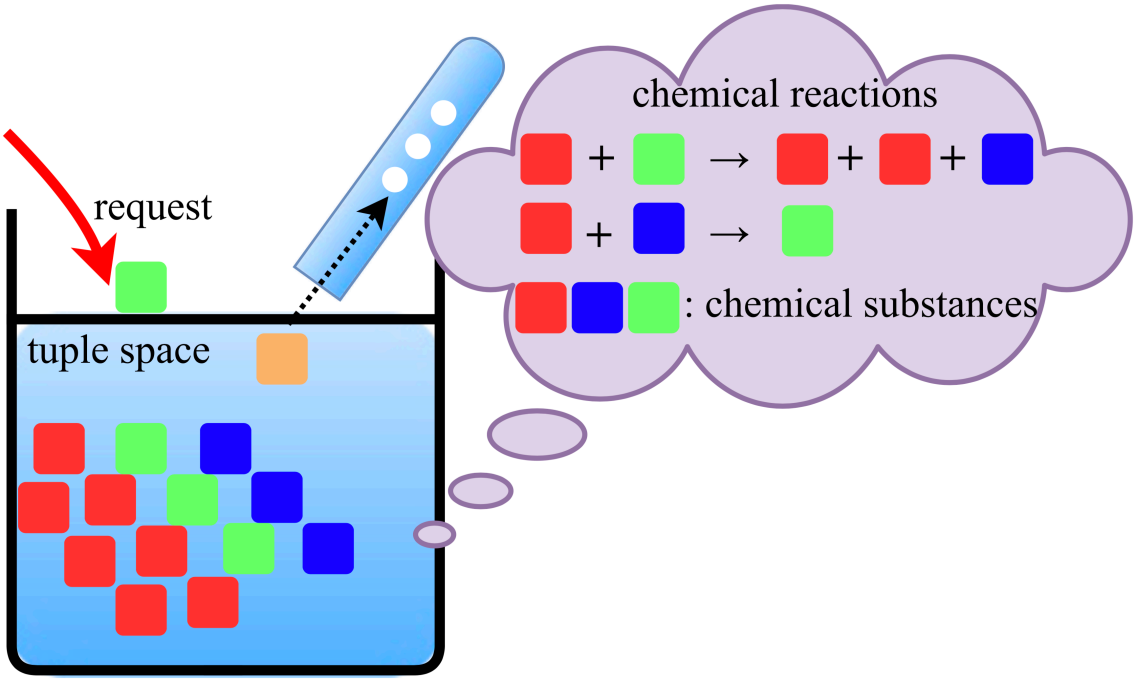


図 2: 生化学反応式を用いたタプル空間モデル [12]

$SERV$ はサービスの需要を表し、その濃度が大きいほどそのサービスに対する需要が多いことを表す。REQ、$toserve(SERV, REQ)$ はそれぞれ、サービス要求、及びサービスの実行結果を表す。化学反応式の矢印の上に書かれた変数は、反応速度係数を表す。反応式 (1) は、サービス需要とそれに対する実行要求が存在する場合に、サービスを実行し、要求を削除すること、及びサービスが成長することを表す。反応式 (2) は、サービス実行要求が存在しないサービスが衰退することを表す。

2.1.2 資源量に基づく制約

反応式 (1) は、反応物の濃度が大きいと、それに応じて反応速度が際限なく大きくなる。しかし、実環境におけるネットワークサービスは、その実行に様々な資源を必要とするため、サービスの実行速度には制約が存在する。そのような環境に適用するために、生化学反応の 1 つである、酵素触媒反応に基づく反応式を用いる [16, 17]。触媒によって、反応そのものには影響を与えず、その濃度によって反応速度を制御することができる [18]。そこで、酵素触媒反応のモデルを用いて、反応式 (1) を以下のように拡張する。

$$SERV|REQ|CATAL \xrightarrow{u} MEDIATE \tag{3}$$

$$MEDIATE \xrightarrow{v} SERV|REQ|CATAL \tag{4}$$

$$MEDIATE \xrightarrow{w} SERV|SERV|CATAL|toserve(SERV, REQ) \tag{5}$$

$CATAL$、$MEDIATE$ はそれぞれ、タプル空間に存在する、サービス実行のために利用可能な資源量、及びサービスに割り当てられる資源量を表す。反応式 (3) 及び (4) は、利用可能な資源量に応じて、サービス資源が割り当てられる様子を表す。反応式 (5) は、サービスに割り当てられた資源量に基づいて、サービスが実行されることを表す。

2.2 NFV への適用

NFV に基づくネットワークにシステムを効率的に運用するためには、各 VNF の配置、サーバ資源の割り当て、及びトラヒック量やサーバ負荷に応じたネットワークフローの経路等を適切に決定する必要がある。そのような挙動を自律分散的に決定するために、上述のモデルを用いる。文献 [11] では、反応式 (1) - (5) における化学物質 $SERV$、REQ、$CATAL$、及び $MEDIATE$ をそれぞれ、VNF、サービスチェイニング要求を持つフローのパケット、サーバにおける利用可能な資源量、及びサーバが提供している VNF に割り当てられた資源量に対応させている。以降では、文献 [11] で提案されている上述のモデルの NFV への適用方法の概略を述べる。

2.2.1 サービスチェイニングの実現

あるフローに対して、そのサービスチェイニング要求により実行される VNF を順に f_1、f_2、f_3…とすると、サービスチェイニング要求 c を式 (6) のように表せる。

$$c \quad = \quad \{f_1, f_2, f_3, \cdots, f_{end}\} \tag{6}$$

サービスチェイニング要求 c を持つフローに対して、VNFf_1 が実行された場合、サービスチェイニング要求 c は式 (7) のように変化する。

$$c \quad \leftarrow \quad c\backslash\{f_1\} = \{f_2, f_3, \cdots, f_{end}\} \tag{7}$$

サービスチェイニング要求 c が次に実行を要求している VNF を $f^1(c)$ と表す。式 (7) の例では、$f^1(c) = f_2$ である。以下では、化学物質の添え字 f はフローを、c はサービスチェイニング要求を表す。

2.2.2 VNF への資源の割り当てと実行

サービスチェイニング要求 c を持ったフローのパケットがサーバへ到着した際、サーバにおいて対応する VNF が存在する場合には、パケットに対して VNF が適用される。このとき、サービスチェイニング要求が 1 つの VNF から構成されている場合には、VNF が適用された後にそのパケットは消失する。一方、サービスチェイニング要求が複数の VNF から構成されている場合には、パケットが持つサービスチェイニング要求から適用した VNF を削除する。このような挙動を反応式 (8) のように表す。

$$VNF_{f^1(c)}|PKT_c \quad \xrightarrow{r_{us}} \quad \begin{cases} VNF_{f^1(c)}|VNF_{f^1(c)}|PKT_{c\backslash\{f^1(c)\}} & \\ |toserve(VNF_{f^1(c)}, PKT_c) & (c\backslash\{f^1(c)\} \neq \emptyset) \\ VNF_{f^1(c)}|VNF_{f^1(c)} & \\ |toserve(VNF_{f^1(c)}, PKT_c) & (c\backslash\{f^1(c)\} = \emptyset) \end{cases} \tag{8}$$

$VNF_{f^1(c)}$ は、サービスチェイニング要求 c が次に実行を要求している VNF を表し、その濃度が大きいほど VNF に対する需要が大きいことを表す。PKT_c は、サービスチェイニング要求が c であるフローを構成するパケットを表す。$toserve(VNF_{f^1(c)}, PKT_c)$ は、サービスチェイニング要求が c であるフローのパケットに対して、次に適用すべき VNF が実行されたことを表す。

また、NFV サーバの資源量とフローの到着レートに応じた VNF への資源割り当てを実現するために、NFV サーバにおいて、以下の化学反応式を導入する。

$$VNF_{f^1(c)}|PKT_c|RSRC_t \xrightarrow{r_{us1}} MEDIATE_{f,t} \tag{9}$$

$$MEDIATE_{f,t} \xrightarrow{r_{us2}} VNF_{f^1(c)}|PKT_c|RSRC_t \tag{10}$$

$$MEDIATE_{f,t} \xrightarrow{r_{us3}} \begin{cases} VNF_{f^1(c)}|VNF_{f^1(c)}|PKT_{c\backslash\{f^1(c)\}}|RSRC_t \\ |toserve(VNF_{f^1(c)}, PKT_c) \quad (c\backslash\{f^1(c)\} \neq \emptyset) \\ VNF_{f^1(c)}|VNF_{f^1(c)}|RSRC_t \\ |toserve(VNF_{f^1(c)}, PKT_c) \quad (c\backslash\{f^1(c)\} = \emptyset) \end{cases} \tag{11}$$

$RSRC_t$ 及び $MEDIATE_{f,t}$ は、サーバ t において利用可能な資源量、及びサーバ t が提供している VNF f に割り当てられた資源量をそれぞれ表す。反応式 (9) 及び (10) は、サービスチェイニング要求 c のフローを構成するパケットが、適用すべき VNF が存在する NFV サーバ t に到着した際に、各 VNF の需要の大きさに応じて資源が割り当てられる様子を表す。反応式 (11) は、VNF f に割り当てられた資源量に基づいて、それが実行されることを表す。

2.3 実システムへの適用

2.2 節で記述した手法を、NFV 環境において実現するための検討を行う。NFV 環境に生化学反応モデルを適用するネットワーク例を図 3 に示す。NFV サーバでは 1 つ、あるいは複数の VNF プログラムが稼働している。NFV サーバにフローのパケットが到着すると、該当する NFV が適用され、NFV サーバから退出する。NFV サーバ上では、前述の生化学反応モデルに基づいて各物質濃度を計算するプログラムが動作しており、NFV サーバへの各フローのパケット到着レートに基づいて、各 VNF に割り当てる資源量を決定する。以降では、本報告において構築した NFV サーバにおける、反応式 (9) - (11) 内の化学物質濃度の設定方法について説明する。

反応式 (9) - (11) における化学物質 VNF、PKT、$RSRC$、$MEDIATE$ の濃度について、実システムでは以下のように設定する。

VNF 初期濃度値を設定し、反応式に応じて更新する。

PKT NFV サーバに到着するパケットの数に応じて決定される。

$RSRC$ サーバにおける CPU 資源量を利用率で表現し、反応式に応じて更新する。

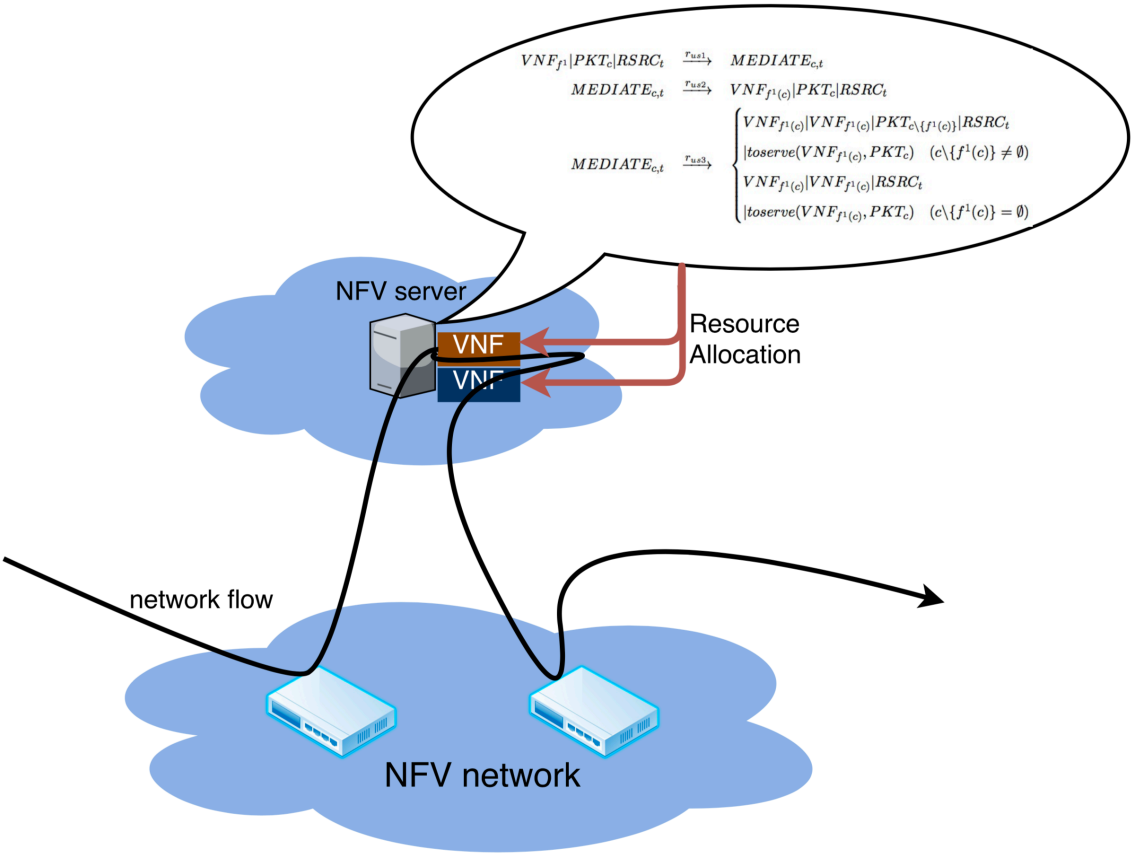


図 3: 生化学反応モデルが適用された NFV 環境 [12]

$MEDIATE$ VNF に割り当てる CPU 資源量を利用率で表現し、反応式に応じて更新する。

VNF には $MEDIATE$ の濃度に応じて CPU 資源を割り当てる。

これらの化学物質の濃度は、文献 [19] に示されている化学反応モデルの数値計算手法に基づき、一定の時間間隔毎に更新される。その間隔で区切られた時間をタイムスロットと定義すると、各タイムスロットの開始時点で、1 つ前のタイムスロットの間に NFV サーバに到着したパケット数に応じて、PKT の濃度を変化させる。次に、タイムスロット内で各化学反応式が実行される回数を反応物の濃度と反応速度係数から算出し、反応処理を実行して各化学物質の濃度を更新する。最後に、VNF に割り当てられる資源量を表す化学物質 $MEDIATE$ の濃度に応じて、VNF に CPU 資源を割り当てる。

3 NFV フレームワーク

NFV の標準化は、European Telecommunications Standards Institute (ETSI) [20] に設立された Industry Specification Group (ISG) [21] によって進められている。ETSI ISG によって発行されている文献 [13] において定義されている NFV のハイレベルフレームワークを図 4 に示す。フレームワークは以下の主要なコンポーネントから構成される。

Virtual Network Functions (VNFs)
: ソフトウェアで実装されたネットワーク機能

NFV Infrastracture (NFVI)
: VNF を実行するためのハードウェア資源 (Hardware resources)、仮想コンピューティング資源 (Virtual Compute)、仮想ストレージ (Virtual Storage)、仮想ネットワーク (Virtual Network)、及びその仮想化を実現する仮想化レイヤ (Virtualisation Layer)

NFV Management and Orchestration (NFV MANO)
: ハードウェア資源、ソフトウェア資源、及び VNF の管理機能とオーケストレーション機能

NFV の実装は、CloudNFV [22] や OpenMANO [23] 等のオープンソースプロジェクトで進められている。本研究では、OPNFV [14] を用いて NFV 環境を実現する。OPNFV は Linux Foundation によって設立されたオープンソースプロジェクトであり、OpenStack [24] 等の既存のオープンソースプロジェクトを統合することで NFV のフレームワーク全体を実装することを目的としている。

OPNFV による NFV 環境は、Kernel-based Virtual Machine (KVM) [25] と呼ばれる Linux カーネル仮想化基盤を用いて実現され、複数の仮想マシンと仮想スイッチから構成される。図 5 に OPNFV によって構築される典型的な NFV 環境を示す。仮想マシンには、アンダークラウド (Undercloud) とオーバークラウド (Overcloud) の 2 種類が存在する。アンダークラウドは OpenStack 環境を構築するために必要となる OpenStack ノードのプロビジョニングや管理のためのコンポーネントが含まれている仮想マシンであり、NFV フレームワークにおけるいずれのコンポーネントにも該当しない。一方、オーバークラウドはアンダークラウドを用いることによって構築される OpenStack プラットフォーム環境であり、コンピュートノード (Compute node) とコントローラノード (Controller node) の 2 種類で構成される。コンピュートノードは、NFV フレームワークにおける NFVI に相当し、VNF はコンピュートノード上の仮想マシンに実現される。コントローラノードは、NFV フレームワークにおける NFV MANO に相当し、OpenStack を用いたコンピュートノード上の仮想マシンのリソース管理や制御、及び OpenDaylight [26] を用いたフロー経路の制御を行う。

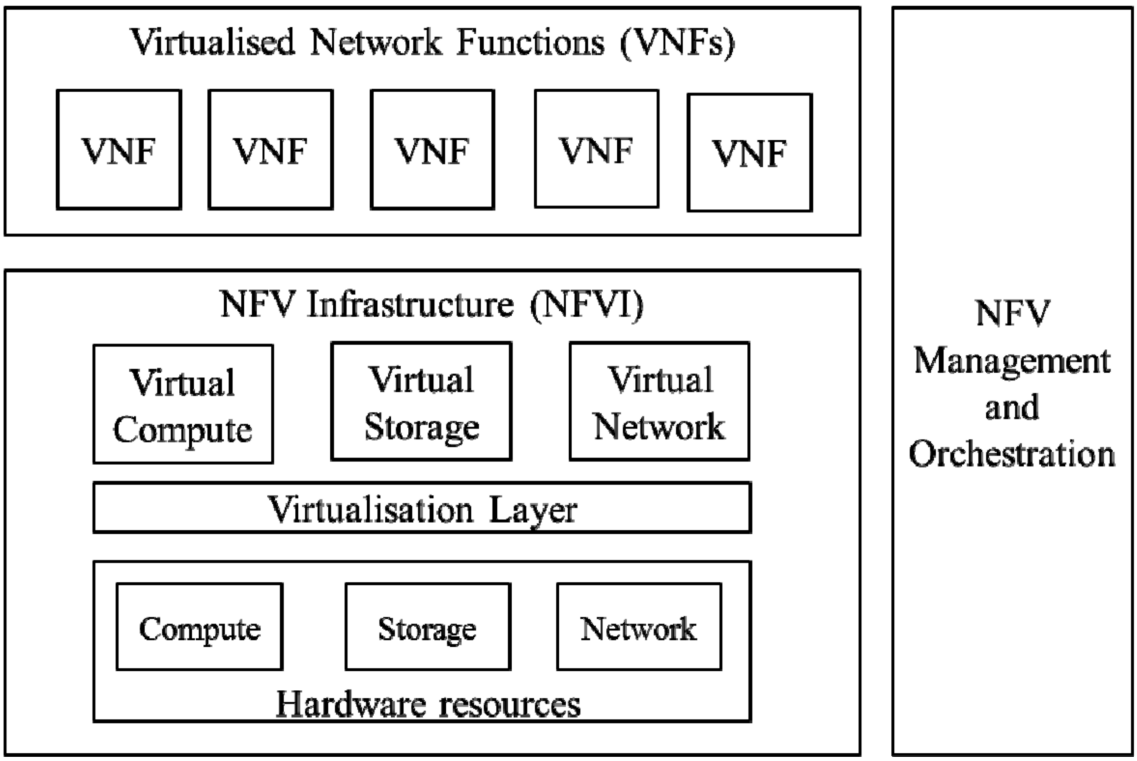


図 4: NFV のハイレベルフレームワーク [13]

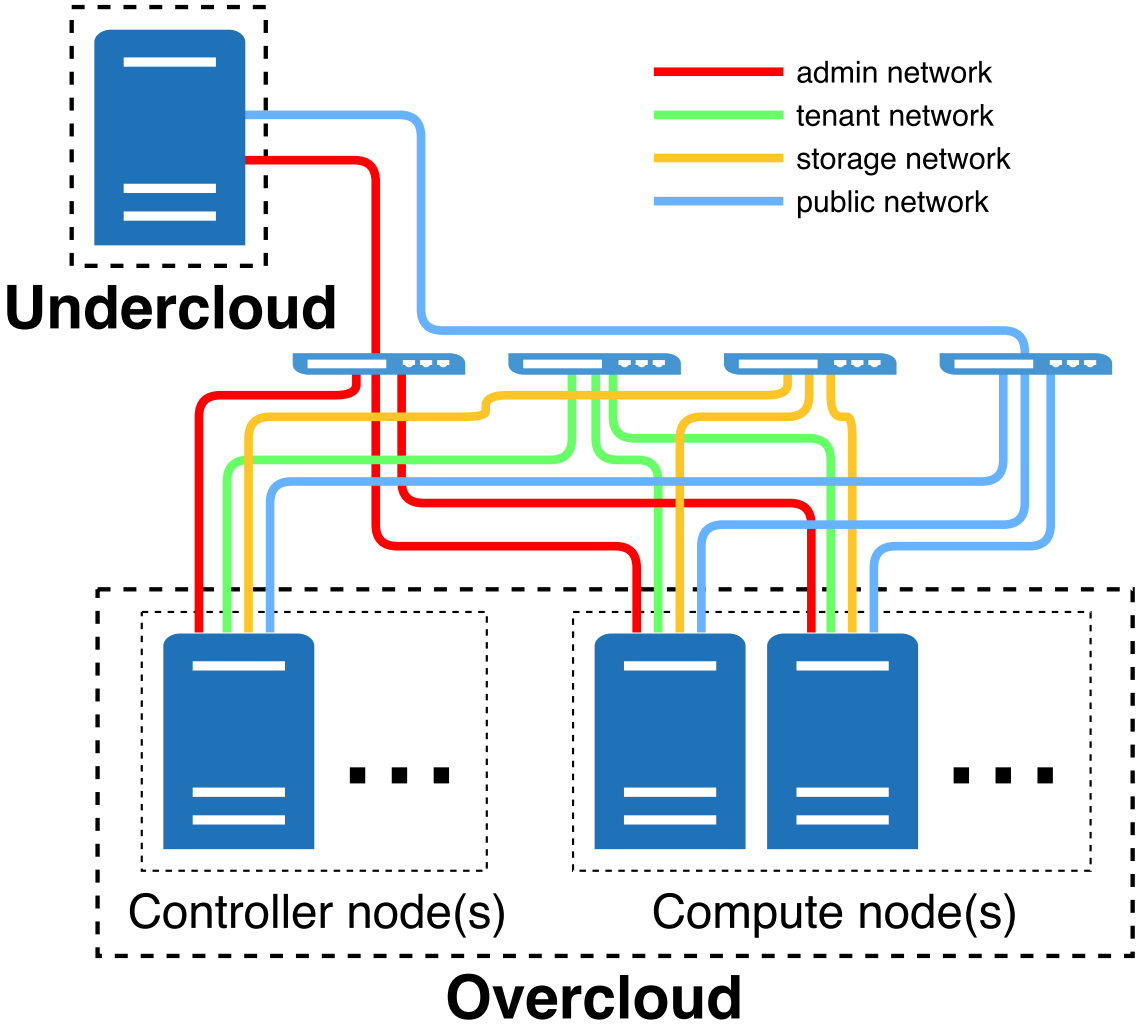


図 5: OPNFV によって構成される典型的な NFV 環境

また、構築される NFV 環境には以下の 4 種類のネットワークが存在し、それぞれに対して 1 つずつ、仮想スイッチである Open vSwitch (OVS) [27] が存在している。

admin network 主にコントロールプレーンが使用するネットワーク

tenant network テナントのトラヒックに用いられるネットワーク

storage network ストレージの I/O の処理に用いられるネットワーク

public network 外部ネットワークに接続するため、及び OpenStack のダッシュボードに接続するためのネットワーク

OpenStack の機能を用いて、コンピュートノード内に仮想マシンを作成すると、コントローラノードとコンピュートノード上に、仮想スイッチや仮想ネットワークが自動的に構成される。作成された仮想マシンは VXLAN [28] を通してコントローラノードからアクセス可能となっており、public network に接続される。

4 実験環境

4.1 実験ネットワーク環境

本研究において構築した実験ネットワーク環境を図 6 に示す。実験ネットワークは、送信ホスト (sender)、受信ホスト (receiver)、NFV サーバ (NFV server) の 3 台の物理サーバによって構成され、1 Gbps のイーサネットによって接続されている。表 1 に、3 台の物理サーバの仕様を示す。

送信ホストは VNF を適用するフローのパケットを送信し、受信ホストはそれを受信する。NFV サーバでは、OPNFV を用いた NFV 環境が構築されている。図 6 においては、Undercloud がアンダークラウドに、Controller0 がコントローラノードに、Compute0 がコンピュートノードにそれぞれ該当する。OVS の仮想スイッチはそれぞれ、br-admin が admin network のために、br-tenant が tenant network のために、br-storage が storage network のために、br-external が public network のために使用される。さらに、コンピュートノード内に VNF サーバとして仮想マシン snort0 が存在する。送受信ホストは共に public network に接続されている。

4.2 VNF プログラム

本報告では、VNF プログラムとして、IDS 機能を持つソフトウェアの 1 つである Snort [15] を利用する。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防止システムであり、IP ネットワーク上での実時間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。

Snort を用いて評価を行うため、VNF サーバである snort0 上に Snort をインストールした。また、VNF サーバで送受信ホスト間の通信を監視するため、仮想スイッチ br-external において、送信ホストから受信ホストへのパケットを、VNF サーバへミラーリングするように設定した。さらに、ICMP パケットを受信した際にアラートを出力するように Snort を設定し、送受信ホスト間の ICMP トラヒックを検出できるようにした。

Snort は、サーバ性能等が原因で処理しきれない量のトラヒックが到着すると、検出能力が低下することが予想される。どのように検出能力が低下するかを把握するために、以下の実験を行った。上述した環境において、パケットサイズを 1,450 Byte、送信間隔を 1 ms、送信パケット数を 10,000 とし、送信ホストから受信ホストへ向けて `ping` コマンドを実行した。その際のパケットのシーケンス番号と Snort での検出時刻の関係を図 7 に示す。図から、全てのパケットが検出されていることがわかる。

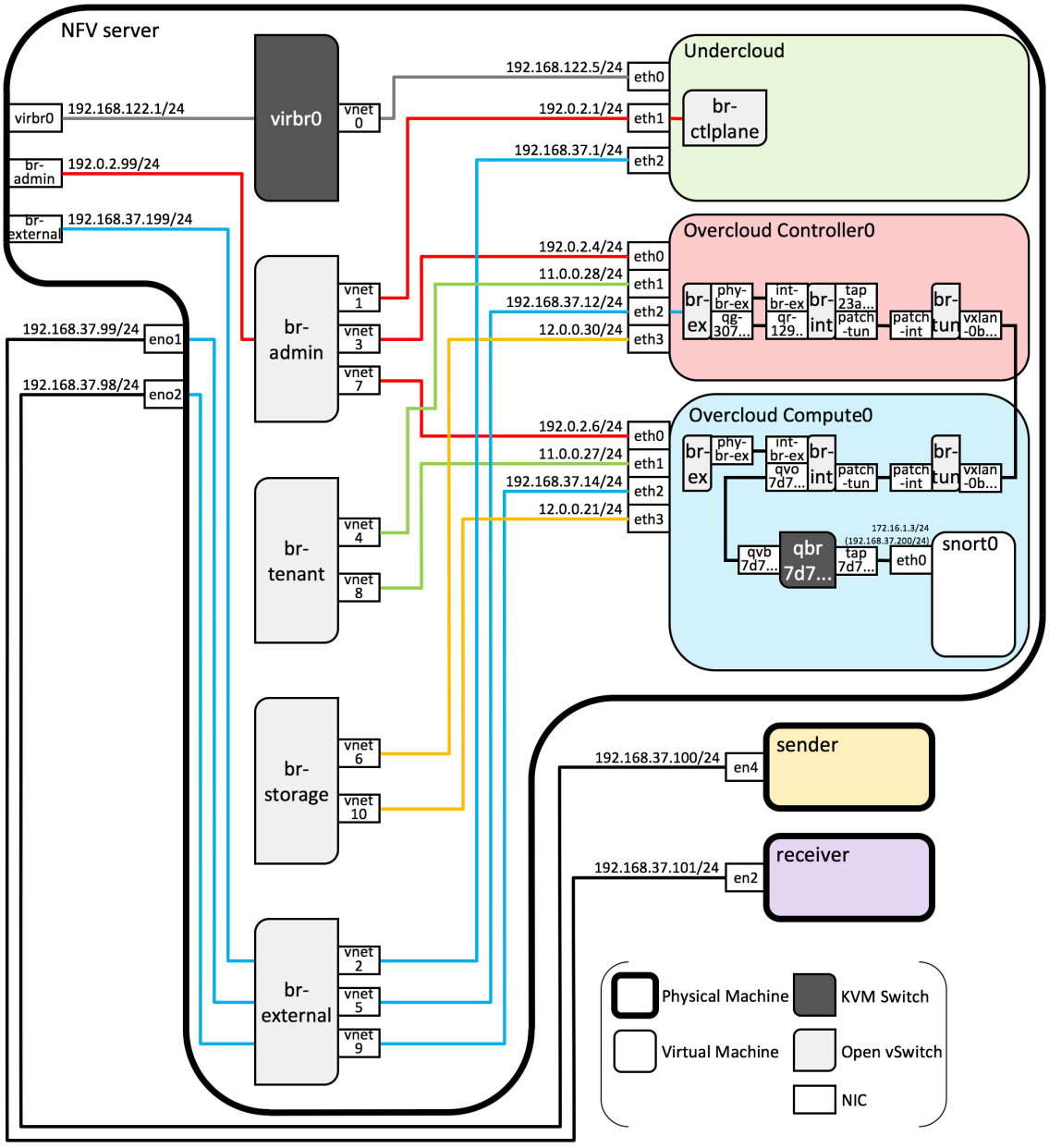


図 6: 実験ネットワーク環境

表 1: 物理サーバの仕様

	送信ホスト	受信ホスト	NFV サーバ
CPU コア数	2	2	16
メモリサイズ	4 GB	4 GB	32 GB
OS	macOS 10.13.2	macOS 10.13.2	CentOS 7.4 1708
カーネルバージョン	17.3.0	17.3.0	3.10.0-693.5.2.el7.x86_64

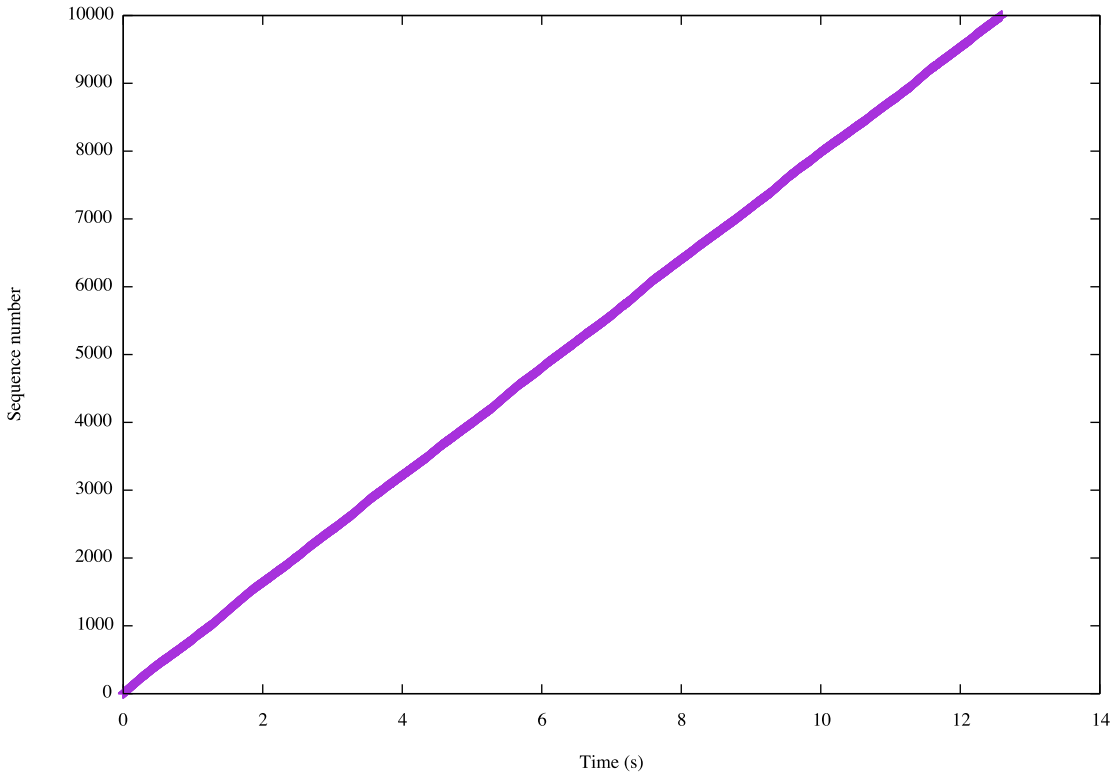


図 7: Snort で検出したパケットのシーケンス番号と検出時刻

次に、VNF サーバの CPU 使用率が制限された環境下における検出性能を評価した。具体的には、KVM で設定することができる仮想マシンのパラメータである cpu_period と cpu_quota を設定することで、仮想マシンの CPU 使用率を制限した。これらのパラメータは、仮想マシンに対する CPU 処理時間の割当上限を設定するものである。それぞれミリ秒単位で設定することができ、cpu_period で指定された期間中に、仮想マシンに割り当てられた総 CPU 時間が cpu_quota の値に達すると、cpu_period の期間が終わるまで、当該仮想マシンにそれ以上 CPU 処理時間が割り当てられなくなる。

cpu_period の値を 100,000 ミリ秒に固定し、cpu_quota を 100,000、50,000、30,000、20,000、10,000、5,000 ミリ秒として同様の実験を行った結果を図 8 に示す。cpu_quota が 30,000 ミリ秒以下の場合に、アラートの出力回数が 10,000 回未満となった。このことから、Snort が使用できる CPU 資源が不足した場合には、検出漏れが発生することが確認できた。また、図から、検出漏れが発生する際には、実験開始後のある時刻から全く検出しなくなるのではなく、断続的に検出漏れが発生していることが分かる。さらに、実験結果を詳細に検証した結果、cpu_quota の値が小さい場合には、実験開始後約 1 秒間は全てのパケットを検出しているが、その後、検知漏れが発生している事がわかった。これらのことから、Snort は以下のように動作していることが予想される。

- 到着したパケットは、検出処理を待つ間、メモリに保存される
- メモリに保存されたパケットは、順に検出処理が行われ、メモリから取り除かれる
- 検出処理速度が、メモリにパケットが蓄積される速度より遅いと、メモリに蓄積されているパケット数が大きくなる
- メモリが一杯の場合、到着したパケットは破棄される

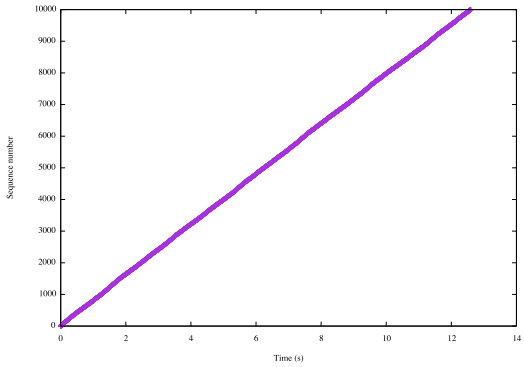
4.3 VNF の性能評価指標

4.2 節で示した Snort の挙動に基づき、Snort の検出性能を評価するための指標として、以下に示す、2 種類の検出率を用いる。

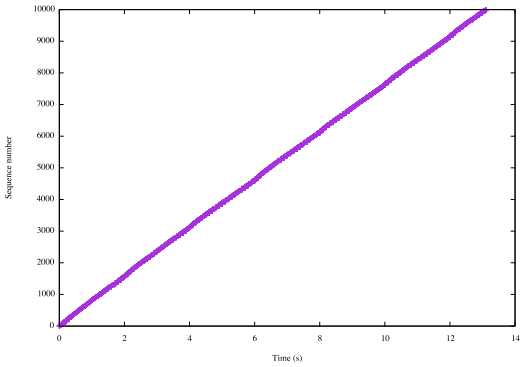
実験時間全体でのパケットの検出率を以下のように定義する。

$$R_{detect}(t)[\%] = \frac{N^*_{detect}(T_{exec}) - N^*_{detect}(T_{ignore})}{N^*_{transmit}(T_{exec}) - N^*_{transmit}(T_{ignore})} \times 100 \quad (12)$$

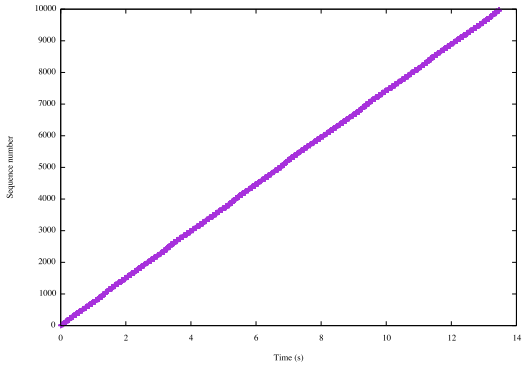
ここで、T_{exec} を実験時間とし、T_{ignore} を実験開始直後の挙動の影響を排除するために設定する、検出率を評価しない時間とし、$N^*_{transmit}(t)$ を実験開始から t 秒後までに送信ホストから送信されたパケット数、$N^*_{detect}(t)$ を実験開始から t 秒後までに Snort がアラートを出力したパケット数とする。



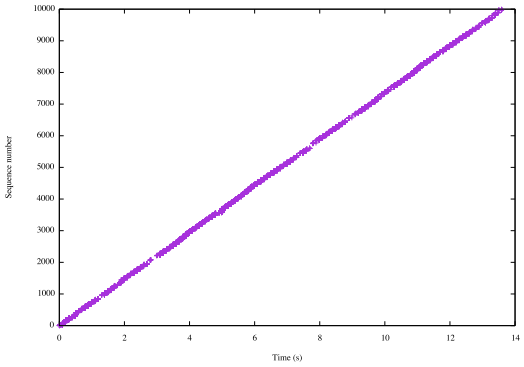
(a) cpu_quota: 100,000



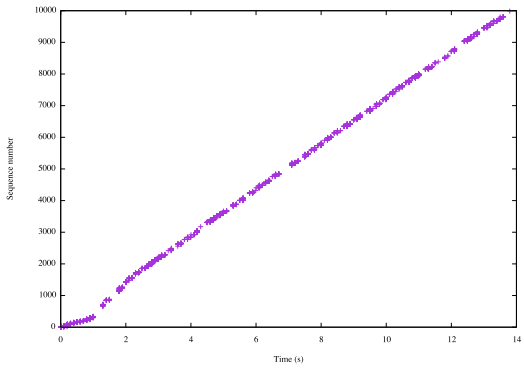
(b) cpu_quota: 50,000



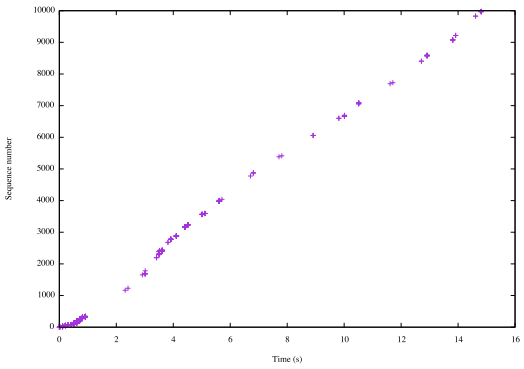
(c) cpu_quota: 30,000



(d) cpu_quota: 20,000



(e) cpu_quota: 10,000



(f) cpu_quota: 5,000

図 8: cpu_quota の値の影響

また、あるタイムスロット t におけるパケットの検出率 $R_{detect}(t)$ を以下のように定義する。

$$R_{detect}(t)[\%] \quad = \quad \frac{N_{detect}(t)}{N_{receive}(t)} \times 100 \tag{13}$$

ここで、$N_{receive}(t)$ は、タイムスロット t の間に controller0 が受信した ICMP パケット数を示す。$N_{detect}(t)$ は、タイムスロット t の間に Snort が出力した ICMP パケットに対するアラートの数を示す。ただし、$N_{receive}(t)$ を記録している controller0 と、$N_{detect}(t)$ を記録している snort0 は異なるため、時刻同期が完全ではない。そのため、$N_{detect}(t)$ を算出する際のタイムスロット t の開始時刻と終了時刻は、実験開始時刻とタイムスロット長を用いて算出する。したがって、$R_{detect}(t)$ は 100%を上回ることがある。

4.4 パケット送信プログラム

送信ホストから受信ホストに向けて ICMP のパケットを任意のレートで送出する方法として簡易なものに、`ping` コマンドを用いる方法がある。本報告においては、フローレートと Snort の検出率の関係を評価するために、以下に示す方法でフローレートを設定することを検討した。

- 1 つの `ping` コマンドによって生成されるフローレートが 5 Mbps となるように、パケットサイズを 1,283 Byte、送信間隔を 2 ms、送信時間を 10 s とする。
- 上記設定の `ping` コマンドを複数個同時に送信端末において実行することによって、全体のフローレートを 5 Mbps から 100 Mbps まで変化させる。
- 受信ホスト、及び VNF サーバにおいては ICMP パケットに対する返答を行わないように設定する。

図 9 に、上述の方法によって送信ホストで実現したフローレートと、送信ホストのネットワークインターフェースカードにおいて `tcpdump` を用いて観察した実際のフローレートとの関係を示す。この結果から、この実験環境においては、上述の方法では設定したフローレートを実現できないことがわかる。また、調査の結果、この現象は `ping` コマンドの実装方法に起因するため、`ping` コマンドを用いることによって任意のフローレートを実現することが困難であることがわかった。

そこで、ICMP パケットを指定したフローレートを実現するための時間間隔で送信するプログラムを作成し、同様の調査を行った。このプログラムは、パケットサイズを 1 KB とし、送信間隔をナノ秒単位で設定することで、設定したレートのトラヒックを実現する。図 10 に、図 9 と同様の実験を行った結果を示す。この結果から、約 600 Mbps 以下のフローレートを正確に実現できることが分かる。

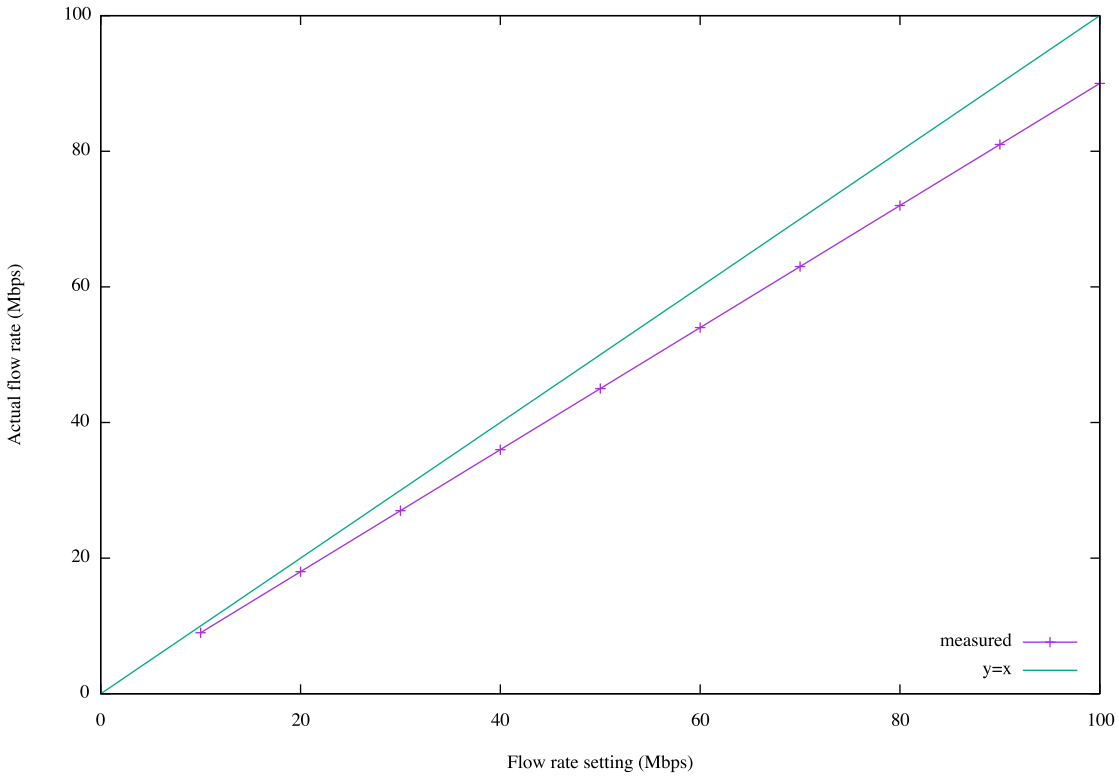


図 9: `ping` コマンドを用いたフローレートの観察結果

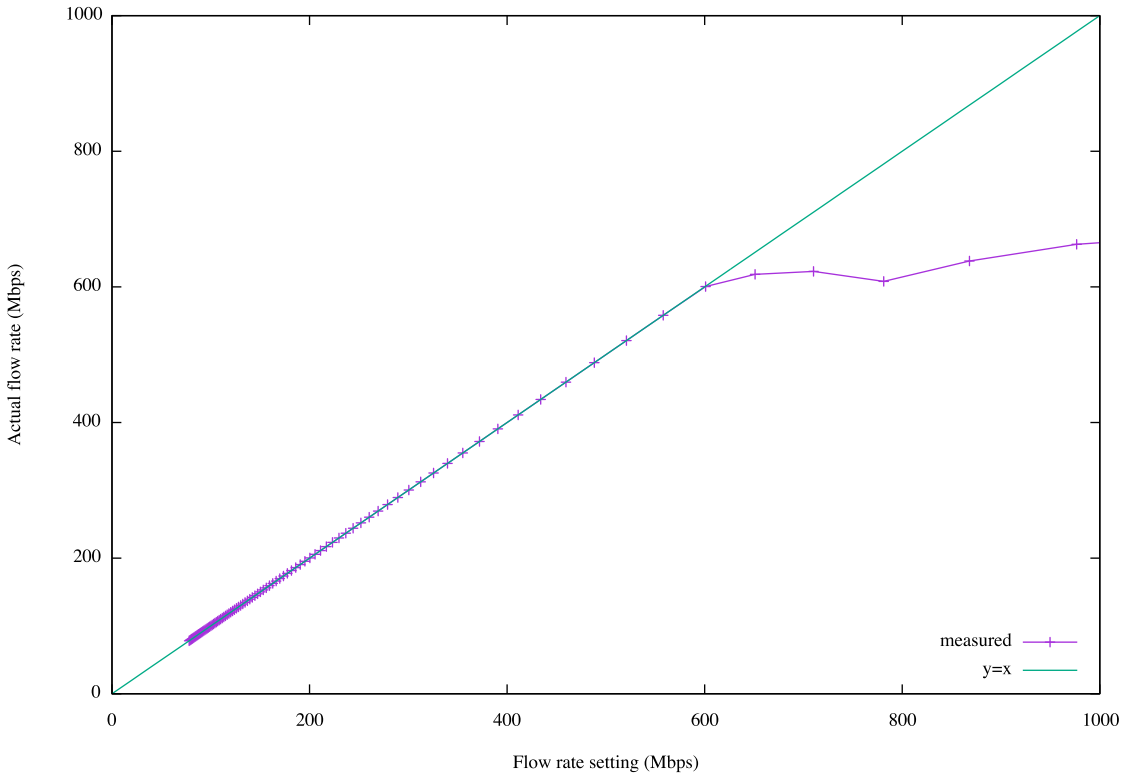


図 10: パケット送信プログラムを用いたフローレートの観察結果

4.5 フローレートの最大値

実験環境において、送信ホストから送信されるパケットのフローレートと、仮想スイッチにおいてミラーリングを行った ICMP のパケットが VNF サーバに正しく到着する割合 (以降では受信率と表記する) の関係を評価した。図 11 に、送信ホストから送出されるフローレートと、VNF サーバにおけるパケットの受信率の関係を示す。図より、フローレートが約 200 Mbps 以上の場合には、VNF サーバのパケット受信率が大きく低下することがわかる。また、実験結果を詳細に検証した結果、フローレートが約 80 Mbps 以上の場合には、受信率が 100%を下回ることがわかった。これらのことから、以降の実験においては、フローレートの最大値を 80 Mbps と設定した。

4.6 提案手法の実現

2 章で述べた手法を実装したプログラムを Controller0 において動作させ、snort0 への CPU 資源の割り当てを行う。4.2 節における実験では、cpu_period と cpu_quota を設定することで VNF サーバに対する CPU 使用率の制限を実現した。しかし、この方法を用いて、CPU 使用率の制限値を変更するためには、VNF サーバを再起動する必要があるため、フローレートの変動に応じて動的に CPU 使用率の制限値を変更することができない。そこで以後の実験では、`cpulimit` コマンドを用いて、Snort のプロセスが用いる CPU 使用率を制限する。

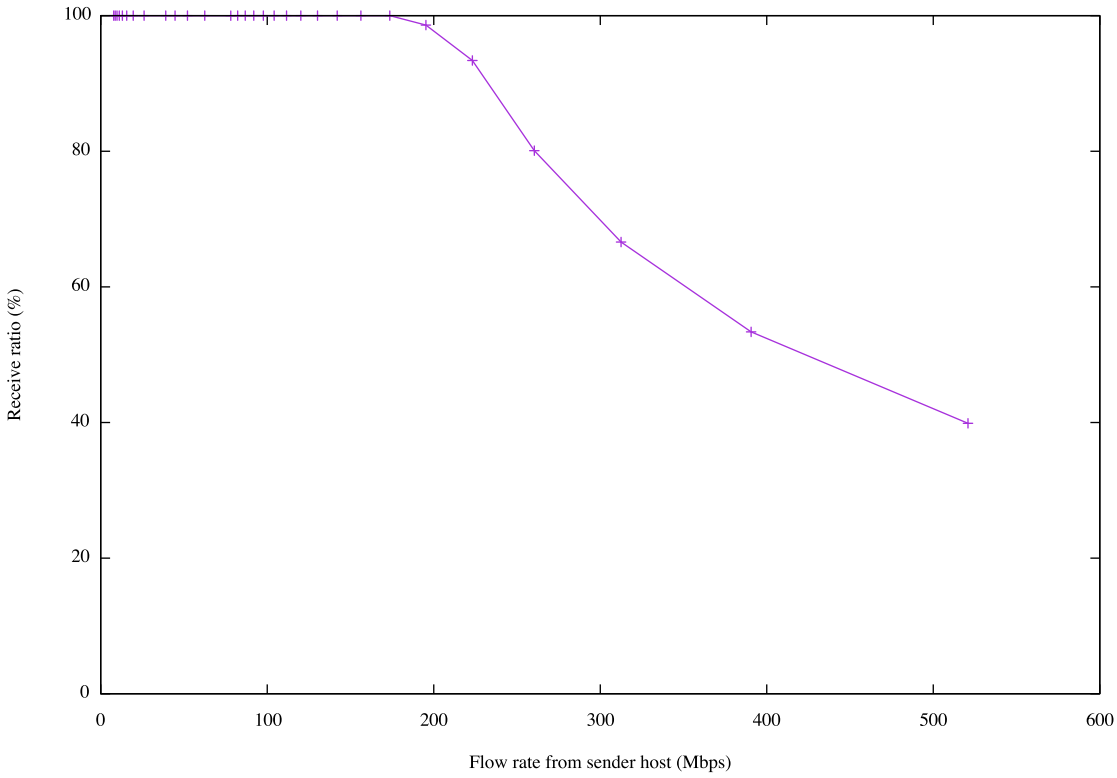


図 11: フローレートと VNF サーバでのパケット受信率

5　実験評価

本章では、4 章で説明した実験環境を用いて実験を行った結果を示し、2 章で述べた手法が正しく動作していることを確認する。

5.1　パラメータ設定

提案方式で用いられる反応式 (9) - (11) における化学物質 VNF、$RSRC$ の濃度の初期値を、それぞれ 2,000、1,000 と設定し、反応速度係数をそれぞれ、$r_{us1} = 0.0001$、$r_{us2} = 0.0001$、$r_{us3} = 0.05$ とした。これらの値は、文献 [11] で行われているシミュレーション評価において用いられているものである。また、タイムスロット長を 1.0 秒と設定した。

$MEDIATE$ の濃度に応じた VNF への CPU 資源の割り当て、及び VNF サーバへのフローレートに応じた化学物質 PKT の濃度の更新を決定するために、フローレートと VNF の CPU 使用率の関係を明らかにする以下の実験を行った。4.4 節で説明したパケット送信プログラムを用いて、送信ホストから受信ホストに向けてトラヒックを送信し、その際のフローレートに対する Snort のプロセスの CPU 使用率を調査した。結果を図 12 に示す。図には、10 回の実験結果の平均値と共に、最大値と最小値をエラーバーを用いてプロットしている。また、フローレートが 60 Mbps 以下の領域で、平均値を基に線形近似を行った結果を合わせて示している。その式は、CPU 使用率を y %、フローレートを x Mbps とすると、以下の式 (14) で示される。

$$y \quad = \quad 1.15 + 1.37 * x \tag{14}$$

以上の実験に基づき、実際のフローレートに応じて、検出漏れを起こさないように、かつ、できるだけ過不足がないように Snort のプロセスの CPU 使用率の制限値を決定するために、以下の式を用いる。ここで、y は Snort の CPU 使用率の制限値 (%)、x はフローレート (Mbps) である。また、図 12 に示すように、Snort 実行時の CPU 使用率にはばらつきがあることから、CPU 資源量に余裕を持たせ、検出漏れを起こさないよう、マージン α を設定する。

$$y \quad = \quad \alpha + 1.15 + 1.37 * x \tag{15}$$

次に、式 (15) を用いて、様々なフローレートを想定して CPU 使用率を制限した環境において、実際のフローレートと式 (12) で求められた検出率の関係を調査した。式 (15) において、$\alpha = 10$ として、x に 10、20、30、40、50、60 をそれぞれ代入して得られた値で Snort のプロセスの CPU 使用率を制限した場合の、実際のフローレートに対する検出率の変化を

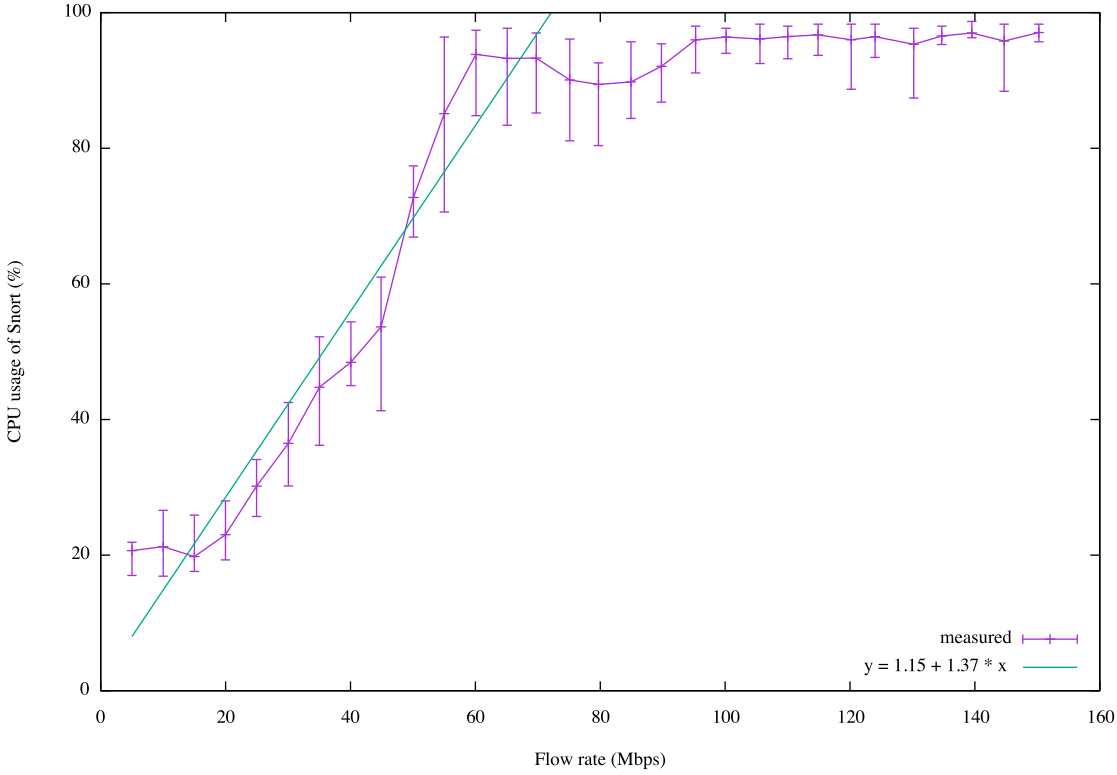


図 12: フローレートと Snort のプロセスの CPU 使用率の関係

調査した。その結果を図 13 に示す。図には、10 回の試行における平均値をプロットしている。この結果から、実際のフローレートが約 50 Mbps 以下であれば、式 (15) を用いて CPU 使用率の制限値を設定することで、過不足のない CPU 資源の割り当てが可能であることが確認できた。

さらに、フローレートの急激な変動やサーバ資源量の減少等の環境変動に備えるために、Snort の CPU 使用率の制限の最大値を 90%とし、それを化学物質 $MEDIATE$ の濃度 1,000 に対応させた。一方、文献 [11] におけるシミュレーション結果より、上述のパラメータを用いた場合に、タイムスロット内で処理することができる化学物質 PKT の濃度の最大値が 45 であることがわかっている。また、式 (15) より、$\alpha = 10$、かつ CPU 使用率の制限値が 90%である時のフローレートは 57.6 Mbps である。このことから、フローレートが 57.6 Mbps の時を、PKT の最大濃度である 45 に対応させた。

5.2 実験内容

本報告においては、ICMP パケットのフローレートが検出率に与える影響を評価する。表 2 に、実験の種類と、設定したフローレートを示す。実験 1 及び 2 においては、実験中にはフローレートを変化させずに、フローレートの違いが与える影響を評価する。実験 3 及び 4 においては、実験中にフローレートを変化させることによって、その変化に提案手法が追随し、Snort の CPU 使用率の制限値が適切に変化し、検出率が維持されることを確認する。

5.3 実験結果と考察

5.3.1 フローレートの影響

図 14 と図 15 に、表 2 における実験 1 と実験 2 の結果をそれぞれ示す。化学物質 VNF、$RSRC$、$MEDIATE$、及び Snort のプロセスへの CPU 使用率の制限値の時間変化をそれぞれ図 14(a) と図 15(a) に、化学物質 $toserve(VNF, PKT)$ の時間変化をそれぞれ図 14(b) と図 15(b) に、また、式 (13) で導出される検出率の時間変化をそれぞれ図 14(c) と図 15(c) に示す。

これらの図から、フローレートに応じて各化学物質濃度が適切に更新され、Snort の CPU 使用率をほぼ過不足なく制限できていることがわかる。図 14(c) から、フローレートが 10 Mbps の場合には、検出率は約 100%で安定していることがわかる。一方、図 15(c) から、フローレートが 50 Mbps の場合には、検出率が 100%に近い値を推移しているが、変動が大きく、検出率が低下する場合があることがわかる。これは、VNF サーバの CPU 使用率の変動による、Snort への CPU 割当量の減少が原因と考えられる。この問題には、式 (15) の α の値

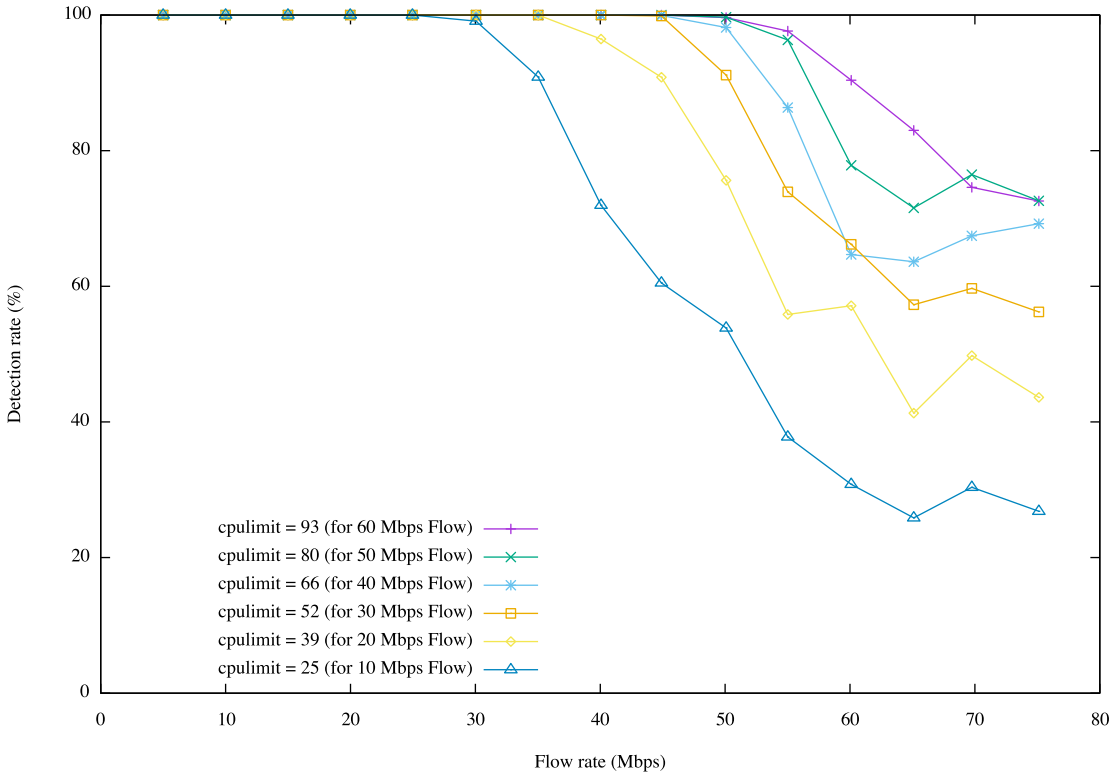


図 13: Snort のプロセスの CPU 使用率を制限した際の検出率の変化

表 2: 実験内容

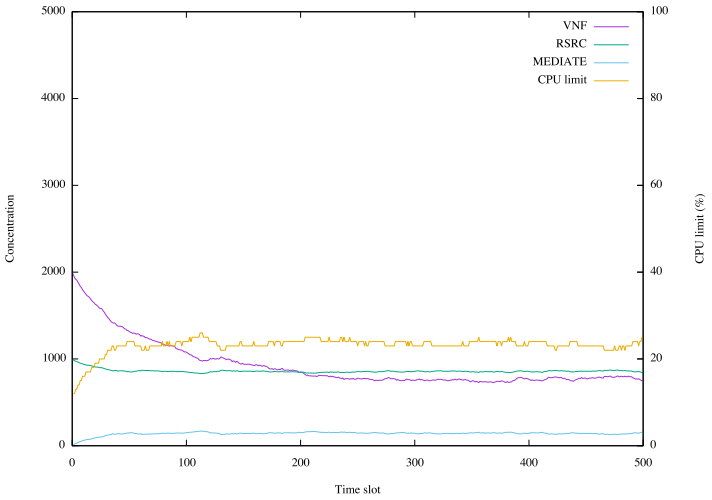
実験	フローレート (Mbps)
1	10 $(0 \leq t < 500)$
2	50 $(0 \leq t < 500)$
3	10 $(0 \leq t < 250)$ 50 $(250 \leq t < 500)$
4	50 $(0 \leq t < 250)$ 10 $(250 \leq t < 500)$

を大きくすることで対処できると考えられるが、α を大きくすることによって、CPU 資源の割り当てに無駄が生じる場合もある。適切な α の設定値については今後の課題としたい。

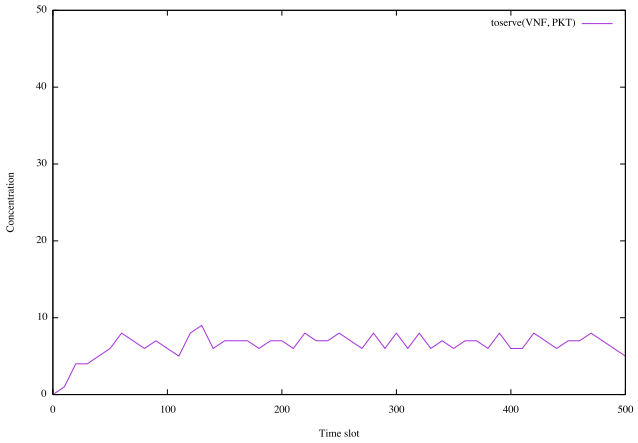
5.3.2 フローレートの変動の影響

図 16 及び図 17 に、表 2 における実験 3 と実験 4 の実験結果をそれぞれ示す。化学物質 VNF、$RSRC$、$MEDIATE$、及び Snort のプロセスへの CPU 使用率の制限値の時間変化をそれぞれ図 16(a) と図 17(a) に、化学物質 $toserve(VNF, PKT)$ の時間変化をそれぞれ図 16(b) と図 17(b) に、また、式 (13) で導出される検出率の時間変化をそれぞれ図 16(c) と図 17(c) に示す。

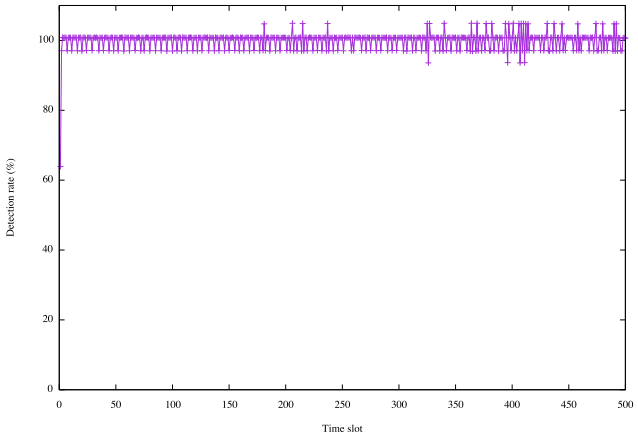
これらの結果から、フローレートが急激に変動する場合においても、各化学物質の濃度が反応式の実行にともなって適切に更新され、Snort の CPU 使用率の制限値が適応的に変化することがわかる。その結果、フローレートの変化にかかわらず、検出率が実験 1 及び実験 2 の結果と同様となることがわかった。この結果により、本報告において NFV フレームワーク上に実装した提案手法が適切に動作することを確認した。



(a) 化学物質濃度と CPU 使用率の制限値の時間変化

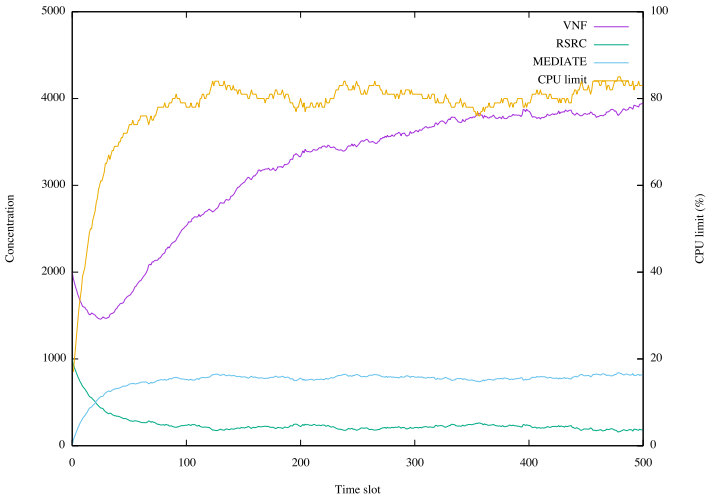


(b) 化学物質 $toserve$ の濃度の時間変化

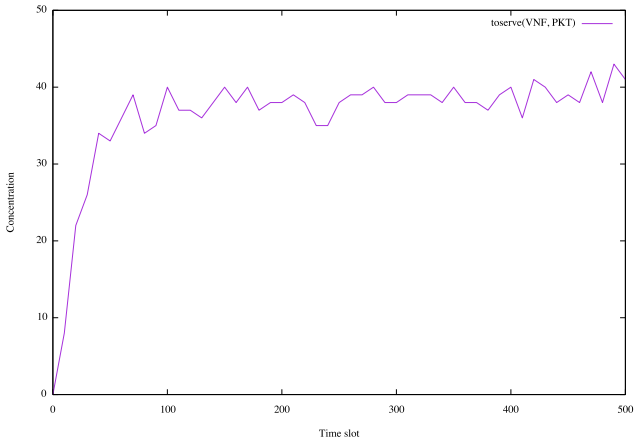


(c) 検出率の時間変化

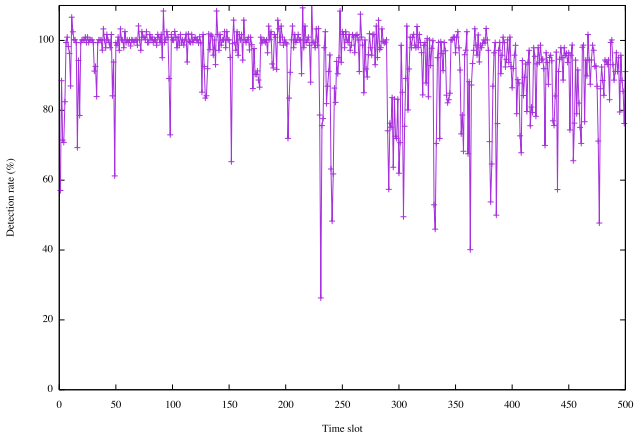
図 14: 実験 1 の結果



(a) 化学物質濃度と CPU 使用率の制限値の時間変化

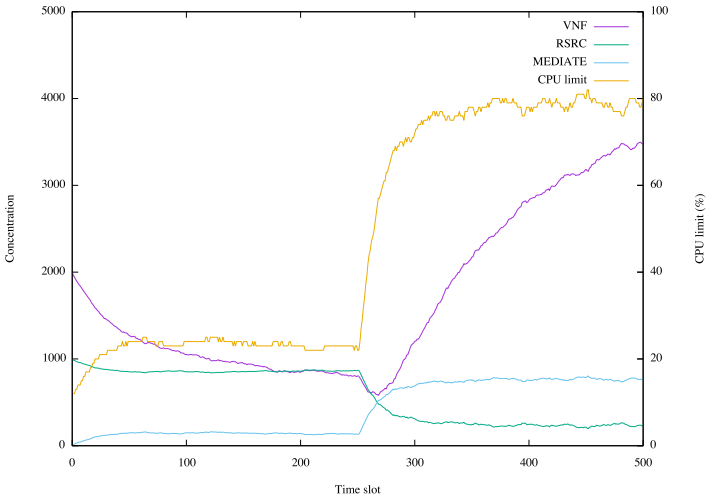


(b) 化学物質 $toserve$ の濃度の時間変化

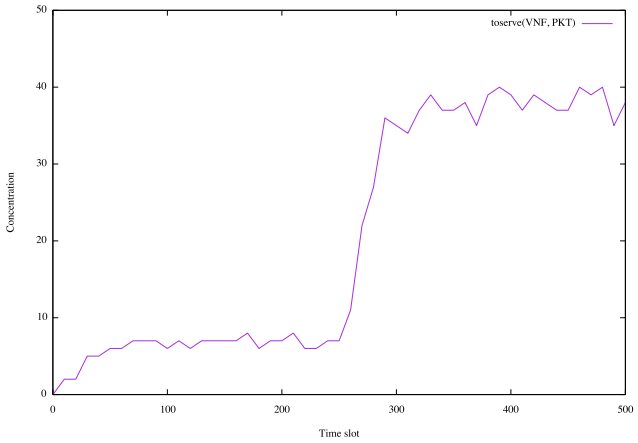


(c) 検出率の時間変化

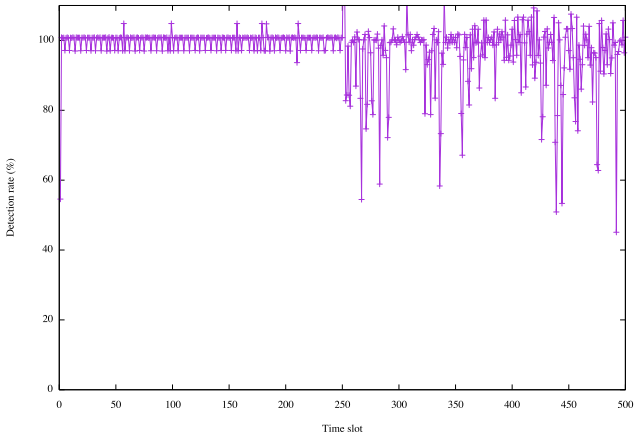
図 15: 実験 2 の結果



(a) 化学物質濃度と CPU 使用率の制限値の時間変化

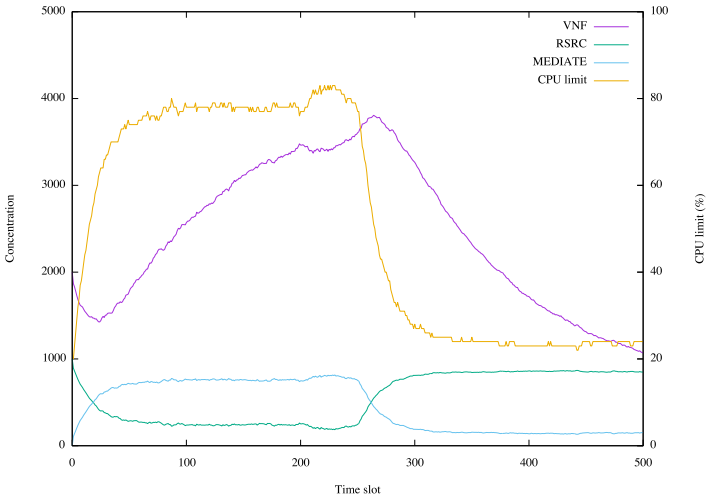


(b) 化学物質 *toserve* の濃度の時間変化

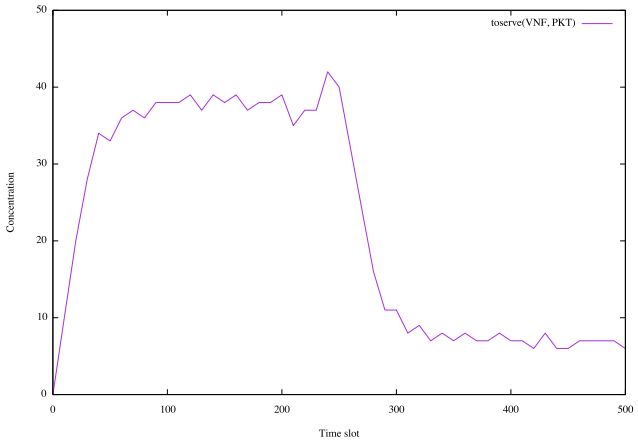


(c) 検出率の時間変化

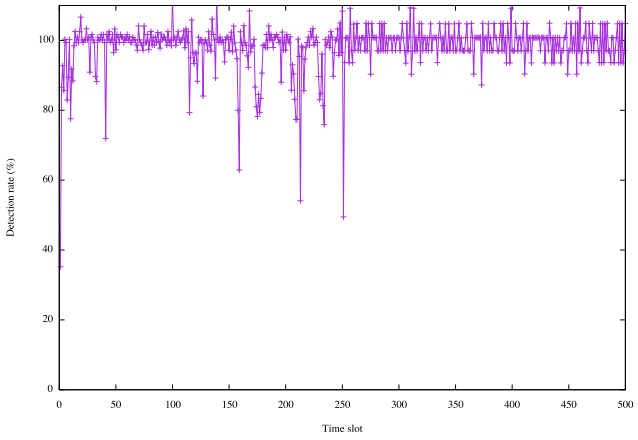
図 16: 実験 3 の結果



(a) 化学物質濃度と CPU 使用率の制限値の時間変化



(b) 化学物質 $toserve$ の濃度の時間変化



(c) 検出率の時間変化

図 17: 実験 4 の結果

6　まとめと今後の課題

本報告では、NFV フレームワークを用いて、生化学反応に基づくサービス空間構築手法を NFV 環境へ適用し、その動作を実験により確認した。具体的には、まず、NFV フレームワークである OPNFV を用いて、NFV 環境を構築した。次に、提案手法に基づいて、ネットワークフローのトラヒック量に応じて、VNF へサーバ資源を割り当てる機構を実現し、NFV フレームワーク上での動作を確認した。様々なシナリオを想定した実験を行った結果、VNF へ CPU 資源を適切に割り当てることにより、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、トラヒック量の動的な変動に対しても、割り当てる CPU 資源量を適応的に調整できることを示した。

今後の課題としては、提案手法のより詳細な実験評価が挙げられる。まず、NFV のサービスチェイニングの実現を行い、文献 [12] において簡易な環境での評価が行われている、複数のフロー及び VNF が存在した場合の実験評価を行いたい。また、文献 [11] で検討されている、複数のタプル空間を接続したネットワークにおける VNF の拡散や移動についても、実験評価を行いたい。

謝辞

本研究を進めるにあたって、たくさんの方々のご協力を頂きました。指導教員である松岡茂登教授には、研究に対する心構えを学ばせて頂き、気を引き締めて研究に望むことができました。感謝申し上げます。また、村田正幸教授のミーティング等での鋭いご指摘によって、研究活動の質を上げることができました。感謝申し上げます。長谷川剛准教授には特にお世話になりました。研究について右も左もわからない自分が、こうして一つの成果を形にすることは、長谷川剛准教授のご指導なくしてはできませんでした。丁寧かつ熱心なご指導の全てに、心より感謝申し上げます。樽谷優弥助教授には、研究室生活を支えて頂き、研究活動への親身なご指導も頂きました。感謝申し上げます。同じ研究室や同じチームの先輩方は、至らない部分も多い自分に、研究活動から研究室生活に至るまで、些細なことから丁寧にご指導して下さいました。感謝申し上げます。また、ここに挙げていない松岡研究室の皆様にも、日頃より様々なご協力を頂きました。感謝申し上げます。最後に、同じ年に同じ研究室に配属された同級生の皆様に、研究の内容は違っても、同じ目標に向かって邁進できたことを感謝申し上げます。

参考文献

[1] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of Things: A Survey on Enabling Technologies, Protocols, and Applications," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 4, pp. 2347–2376, Fourthquarter 2015.

[2] ESTI, "Network Functions Virtualisation - White Paper1." available at `https://portal.etsi.org/nfv/nfv_white_paper.pdf`.

[3] K. Ingham and S. Forrest, "A History and Survey of Network Firewalls," tech. rep., University of New Mexico, Jan. 2002.

[4] M. Finsterbusch, C. Richter, E. Rocha, J.-A. Muller, and K. Hanssgen, "A Survey of Payload-Based Traffic Classification Approaches," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 2, pp. 1135–1156, Oct. 2013.

[5] D. Wing, "Network Address Translation: Extending the Internet Address Space," *IEEE Internet Computing*, vol. 14, no. 4, pp. 66–70, June 2010.

[6] M. Olsson, S. Rommer, C. Mulligan, S. Sultana, and L. Frid, *SAE and the Evolved Packet Core: Driving the mobile broadband revolution.* Academic Press, Aug. 2009.

[7] ETSI, "Network Functions Virtualisation - White Paper2." available at `https://portal.etsi.org/nfv/nfv_white_paper2.pdf`.

[8] 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 International Workshop on Quality of Service*, pp. 27–36, June 1999.

[9] M. Viroli, M. Casadei, S. Montagna, and F. Zambonelli, "Spatial Coordination 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.

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

[11] K. Sakata, “Adaptive and autonomous placement method of virtualized network functions based on biochemical reactions,” Master’s thesis, Department of Information Networking Graduate School of Information Science and Technology Osaka University, Feb. 2018.

[12] 黒川 稜太, **生化学反応モデルに基づいた仮想ネットワーク機能の動的資源配分手法の実験評価**. 特別研究報告, 大阪大学 基礎工学部 情報科学科, Feb. 2017.

[13] 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`.

[14] “OPNFV: Home.” available at `https://www.opnfv.org`.

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

[16] K. A. Johnson and R. S. Goody, “The Original Michaelis Constant: Translation of the 1913 Michaelis–Menten Paper,” *Biochemistry*, vol. 50, no. 39, pp. 8264–8269, Sep. 2011.

[17] R. N. Goldberg, Y. B. Tewari, and T. N. Bhat, “Thermodynamics of enzyme-catalyzed reactions,” *Bioinformatics*, vol. 20, no. 16, pp. 2874–2877, May 2004.

[18] W. W. Cleland, “The kinetics of enzyme-catalyzed reactions with two or more substrates or products,” *Biochimica et Biophysica Acta (BBA) - Specialized Section on Enzymological Subjects*, vol. 67, pp. 173–187, May 1963.

[19] H. Li, Y. Cao, L. R. Petzold, and D. T. Gillespie, “Algorithms and Software for Stochastic Simulation of Biochemical Reacting Systems,” *Biotechnology Progress*, vol. 24, no. 1, pp. 56–61, Feb. 2008.

[20] “ETSI - Welcome to the World of Standards!.” available at `http://www.etsi.org`.

[21] “ETSI - Industry Specification Groups (ISGs).” available at `http://www.etsi.org/about/how-we-work/how-we-organize-our-work/industry-specification-groups-isgs`.

[22] R. Mijumbi, J. Serrat, and J.-L. Gorricho, "Network Function Virtualization: State-of-the-art and Research Challenges," *IEEE Communications Surveys & Tutorials Tutorials*, vol. 18, no. 1, pp. 236–262, Sep. 2015.

[23] R. Mijumbi, J. Serrat, J.-L. Gorricho, S. Latré, M. Charalambides, and D. Lopez, "Management and Orchestration Challenges in Network Function Virtualization," *IEEE Communications Magazine*, vol. 54, no. 1, pp. 98–105, Jan. 2016.

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

[25] S. G. Soriga and M. Barbulescu, "A comparison of the performance and scalability of Xen and KVM hypervisors," in *Proceedings of 2013 RoEduNet International Conference 12th Edition: Networking in Education and Research*, Sept. 2013.

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

[27] "Open vSwitch." available at `http://www.openvswitch.org`.

[28] M. A. Nadeem and T. Karamat, "A survey of cloud network overlay protocols," in *Proceedings of 2016 Sixth International Conference on Digital Information and Communication Technology and its Applications (DICTAP)*, pp. 177–182, July 2016.