情報処理学会論文誌 Vol.59 No.2 528–543 (Feb. 2018)

OpenFlowを用いた医療情報ネットワークの設計と実装

小野 悟[1,a)] 岩崎 裕輔[2,b)] 峰野 博史[1,c)] 猿渡 俊介[2,d)] 渡辺 尚[2,e)]

受付日 2017年4月24日, 採録日 2017年11月7日

概要：現在の医療情報ネットワークでは，増大するスループットとロバスト性への要求に対応するためにコアスイッチが高価格化の一途をたどっているという問題がある．このような問題を解決するネットワークシステムとして，本稿では OpenFlow を用いた医療情報ネットワーク「MediFlow」を提案する．MediFlow では，高価かつ高性能なコアスイッチを，フルコネクトされた複数の安価な OpenFlow スイッチで構成される MediFlow コアネットワークで置き換える．MediFlow コアネットワークを用いることで，医療情報ネットワークにおける時間的なトラヒックの偏りに対して適応的に通信容量を割り当てることができる．リンクに障害が発生した場合でも，動的に経路を割り当てることで通信のロバスト性を提供する．MediFlow を市販の OpenFlow スイッチに実装した結果，複数のフローが発生して輻輳が発生してから約 7 秒後に各フローに個別の経路を割り当ててスループットを向上させることができることが確認できた．また，フローが利用している経路中でリンク断が発生した場合でも，約 0.7 秒の切断時間で経路が再設定されることが確認できた．シミュレーションによる評価の結果，MediFlow を用いて現状のコアスイッチを用いた場合に漸近する性能を発揮できることも確認できた．

キーワード：医療情報，SDN，OpenFlow，ネットワーク管理，実機実装

Network Virtualization for Scalable Medical Information Networks

Satoru Ono[1,a)] Yusuke Iwasaki[2,b)] Hiroshi Mineno[1,c)] Shunsuke Saruwatari[2,d)] Takashi Watanabe[2,e)]

Received: April 24, 2017, Accepted: November 7, 2017

Abstract: In the current medical information networks, there is a problem that the expense of core switches is increasing to meet demand for throughput and robustness. To solve the expense problem, this paper proposes MediFlow, a medical information network using OpenFlow. MediFlow replaces costly and high performance core switches with a MediFlow core network which consists of multiple inexpensive fully connected OpenFlow switches. The MediFlow core network adaptively allocates communication capacity to time-varying traffic in a medical information network. Even when a link failure occurs, MediFlow dynamically assigns a route to provide communication robustness. We implemented MediFlow using commercial OpenFlow switches. The experimental evaluation shows that MediFlow improves throughput by allocating individual routes to each flow about 7 seconds after congestion occurred by converging multiple flows. Additionally, even if a link failure occurred in the route used by the flow, the MediFlow re-establishes a route with a disconnection time of about 0.7 seconds. The simulation results also shows that MediFlow provides the asymptotic performance as the current core switch.

Keywords: medical information system, SDN, OpenFlow, network management, implementation

1. はじめに

医療情報分野では，この四半世紀中で大きな変革があっ

[1] 静岡大学創造科学技術大学院
Graduate School of Science and Technology, Shizuoka University, Hamamatsu, Shizuoka 432–8561, Japan
[2] 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University, Suita, Osaka 565–0871, Japan
[a)] takumi@hama-med.ac.jp
[b)] iwasaki.yusuke@ist.osaka-u.ac.jp
[c)] mineno@inf.shizuoka.ac.jp
[d)] saru@ist.osaka-u.ac.jp
[e)] watanabe@ist.osaka-u.ac.jp

た．古くは汎用機を用いたレセコンと呼ばれる診療報酬請求明細書の計算と出力のために整備されていたコンピュータの活用は事務分野が中心であったが，ここ十数年の間に医療現場そのものまでに電子化が波及するようになった．平成 13 年度の e-Japan 構想に合わせて策定された厚生労働省の「保健医療分野の情報化にむけてのグランドデザイン」[1] では，その達成目標として，電子カルテに関しては平成 18 年度までに 400 床以上の病院 6 割以上に普及・全診療所の 6 割以上に普及を，レセプト電算処理システムに関しては平成 18 年度までに全国の病院 7 割以上に普及させることを掲げていた．これらの目標を受けて，400 床以上の病院における電子カルテの普及率は平成 27 年現在で 70.1%を達成した [2]．電子カルテを導入することにより，ほぼすべての診療記録が電子化されることとなる．

このような電子化の流れの中で，医療情報ネットワークを流通するネットワークコンテンツは年々肥大化している．診療現場である外来ブースや，病棟のスタッフステーションには多くの端末が設置されている．病床数が千を超える特定機能病院では数千台規模の端末が常設されることも多い．検体検査，処方，X 線写真，生理検査等あらゆる医療行為は，絶えずこれらの端末から電子情報としてオーダされるため，受け手となる部門システムに確実に送信されなければならない．検体検査や X 線写真，生理検査等部門システムからの結果返信が必要な場合には，即座にこれらの情報を送り返す必要がある．特に外来ブースではこれらの送受信に係るレスポンスが患者の待ち時間や医師の業務効率に大きな影響を与える．

このような医療情報ネットワークにおいて，ネットワークシステムの高価格化が問題となっている．高価格化の要因は，医療情報ネットワークにおけるトラヒックの時変動性とロバスト性への要求の 2 つである．この 2 つの要因を，医療情報ネットワーク特有の「端末側のアプリケーションの変更をともなうアプローチがとれない」という制約下で対応しなければならない．これらの要因と制約に関しては 2 章で詳細に議論する．

以上の観点から，本稿では，OpenFlow [3], [4], [5] を用いた医療情報ネットワーク「MediFlow」を提案する．MediFlow では，現在の医療情報ネットワークの高価なコアスイッチを複数の安価な OpenFlow スイッチで構成する MediFlow コアネットワークで代替する．時間帯によって特性の異なるトラヒックに対して経路を動的に割り当てることで各時間帯で通信の多いフローに通信容量を提供できる仕組みを実現する．また，リンクに障害が発生した場合でも障害の発生したリンクを迂回するように経路を割り当てることでロバスト性を確保する．

MediFlow を実際の OpenFlow スイッチを用いて実装した結果，複数のフローが存在する場合に各フローに経路を動的に割り当てることができることを確認できた．また，リンク断が発生した場合でも通信を継続して行えることも確認できた．現在の医療情報ネットワークのトポロジを模した計算機シミュレーションによる評価では，MediFlow は現状のコアスイッチに漸近する性能を発揮できることも分かった．

本稿の構成は以下のとおりである．2 章では，現在の医療情報ネットワークの課題と制約について，トラヒックの特徴とロバスト性の要求の 2 つの側面から議論する．3 章では，提案手法の詳細について述べる．続く 4 章では，実機の評価環境について述べ，提案方式である MediFlow を実機を用いて評価を行う．5 章では，MediFlow のスケーラビリティをシミュレーションによって評価する．6 章では関連研究について議論する．7 章では提案手法の適用可能性およびスイッチの価格差要因に関して議論する．最後に 8 章で本稿のまとめを行う．

2. 医療情報ネットワークの課題と制約

2.1 トラヒックの時変動性による高価格化

増加するスループットに対する要求の中で，医療情報ネットワークのトラヒックの時変動性がネットワークシステムの高価格化を招いている．医療情報ネットワークを流通するトラヒックはバースト的であり，ピーク時の性能が常時要求されることは少ない．**図 1** に，H 大学病院で実際に稼働する医療情報ネットワークで収集したトラヒックを示す．Simple Network Management Protocol（SNMP）の RMON によるモニタは監視対象の装置に対する負荷が大きく，リアルタイム解析が困難であったため，Network Time Machine [6] を用いてトラヒック収集を行った．

Network Time Machine には，1 Gbps の Small Form-factor Pluggable（SFP）モジュールを最大で 4 つ搭載可能である．それぞれのモジュールから異なるセグメントのトラヒックを同時に収集することができる．3 TB のストレージを備えているので，長時間の連続した収集が可能で

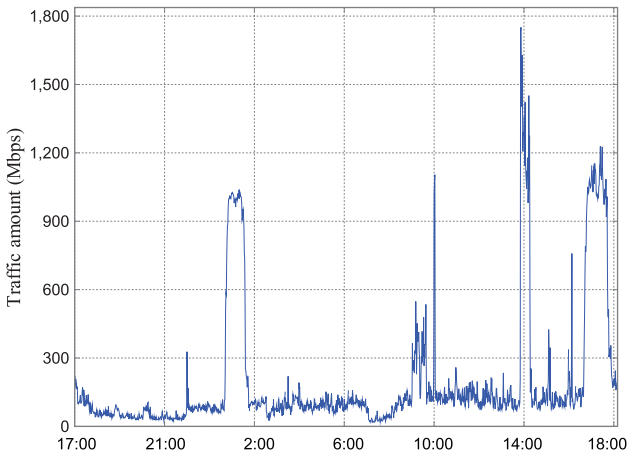


図 1 医療情報ネットワークのトラヒック推移
Fig. 1 Traffic trend on a medical information network.

ある．今回の計測では，診療業務が救急患者の受入や，病棟での入院患者への対応業務によって 24 時間継続して行われていることから，業務全体の傾向を把握するために午後 5 時から翌日の午後 6 時までの時間帯で収集した．収集箇所は，電子カルテシステムのサーバが接続されている 4 台のサーバスイッチとコアスイッチがリンクされているポートである．コアスイッチには，院内すべての端末を収容する多くのエッジスイッチや，無線 AP が収容されている PoE スイッチが接続されているため，電子カルテを閲覧するためのトラヒックはこれらのポートを監視することですべて捕捉可能である．

図 1 より，医療情報ネットワークでは帯域を定常的に消費しているのではなく，偏差が大きく時間帯によって特性が異なるトラヒックであることが分かる．図 1 では，午前 0 時頃と 10 時頃，午後 2 時頃と 16 時頃の 4 つのピークが確認できる．このうち，午前 0 時頃に発生して約 2 時間継続している 1 Gbps 前後のピークはバックアップトラヒックである．その他のピークは業務トラヒックと推察する．最大ピークは 1.8 Gbps 程度であるが，ピーク外の時間帯はおおむね 300 Mbps 未満のトラヒックで推移している．

時間帯による特徴が分かりやすい例としては，図 1 中の午前 0 時頃に発生しているバックアップのトラヒックがあげられる．東日本大震災を契機に，災害下でも医療情報システムの継続性を保つための全国国立大学病院診療情報バックアップ事業（The Gemini Project）が平成 25 年度より開始されている [7]．各国立大学病院では，このバックアップの対象として，2 つのソースを広域ネットワークを介して保存する必要がある．1 つめは災害復旧時に利用するプロプライエタリなシステムファイルで 2 つめは事業に参加する全施設が災害時に共通して参照可能な標準化型ファイルである．これらのソースは一様に大容量であるため，ネットワークが高帯域に長時間専有されてしまう．

このようなピーク時の性能確保のために，現状ではネットワーク機器として高価格なものを用いなければならない．これまでの医療情報ネットワークでは，あらかじめ特定の宛先にトラヒックが集中することを予測し，その経路が含まれるリンクを Link AggreGation（LAG）等を用いて多重化していた．このような環境下では，トラヒックがバースト的である場合，多重化された帯域は常時利用されず，LAG 化されたポートは利用の有無にかかわらず常時占有されてしまうという欠点がある．さらにバックアップタスクの存在はネットワーク機器の高価格化の一因でもある．バックアップタスクは綿密に実行のタイミングを計画しているものの，バックアップのソースは経年肥大するため，タスクスケジュールの定期的な見直し等，全体設計を含めてコスト増加の要因となっている．

表 1 機器別の障害発生件数の比率
Table 1 Failure rate for each devices.

デバイス名	比率
laptop	47%
printer	22%
desktop	18%
PDA	10%
storage	1%
server	1%
network	1%

2.2 ロバスト性の要求による高価格化

医療情報ネットワークでは，端末が多量に導入されるに従って相対的にネットワーク機器の故障率は低くなっているにもかかわらず，ロバスト性への対応がネットワーク機器の高価格化を招いている．近年では，医療分野においてもスマートフォンやタブレット等多様な端末が利用されるようになっている [8]．これらの端末は可搬性が高いため，ベッドサイドでの入力事象発生時の発生源入力をはじめ，各種医療行為に対する 3 点認証（患者，医療行為，施行者）への活用等に用いられていることから，トラヒックに対する高いロバスト性が求められている．電子カルテシステムでは，端末だけでなく多種の機器が稼働している．電子カルテシステムの主要構成要素の障害は人命に影響を及ぼす可能性があることからも，高いロバスト性が要求されている．

現状の医療情報ネットワークでは，システムへのロバスト性に対して高価な機器や多くのマネジメント費用で対応している．ロバスト性の向上を図るために，現状では主要構成要素を冗長化している．冗長化によって全体のロバスト性は向上するが，反面で高額な費用が必要となる．特に，ネットワークの中心となるコアスイッチは高性能化の一途をたどっており，シャシー型コアスイッチを用いる等大きなコスト増加の要因となっている．

しかしながら，近年は構成機器の信頼性が向上したことで相対的にネットワーク機器の故障率が低下しているため，これら冗長構成が有効活用される場面は少ない．**表 1** に平成 28 年度の H 大学病院の医療情報トータルシステムにおける機器別障害発生件数の比率を示す．ラップトップ型コンピュータの障害が最も多く，プリンタの障害件数がこれに次ぐ．これに対し，サーバやネットワーク等主要構成要素の故障が非常に少ないことが分かる．たとえば，Virtual Router Redundancy Protocol（VRRP）等を用いて冗長構成とした場合には，通常バックアップ側の機器は稼働しないため，マスタ側の機器に障害等が発生しない限りバックアップ側が活用されることはない．シャシー型スイッチで装置内冗長構成をとった場合，CISCO L3 スイッチでのスーパバイザ冗長化では一方のエンジンは停止している．両エンジンは同時に稼働しない．すなわち，現在の

医療情報ネットワークでは，高価かつ高性能である機器の性能が十分に活用されていない．

2.3 医療情報ネットワークの制約

2.1 節，2.2 節で述べたように，医療情報ネットワークの課題はトラヒックの時変動性とロバスト性の要求に対応するためにネットワーク機器が高価格になっていることである．このような課題に対して，e-Science やデータセンタネットワーク等で検討されているような端末側に MPTCP の機能を持たせたりルートを選択可能な機能を持たせたりすることができれば解決できる可能性がある [9], [10].

しかしながら，MPTCP が機能するためには，端末側だけでなく，サーバ側でも有効にする必要がある．国内大手のエンタープライズ向け電子カルテシステムのソリューションでは，日本電気，富士通，日立等がよく知られているが，いずれもサーバ OS は Windows である．本稿執筆時点において MPTCP をサポートする OS としては，サーバ環境では Linux および Free BSD，クライアント環境では，Mac OS X および iOS である．すなわち，現時点ではエンタープライズ向け電子カルテシステムにおいて，MPTCP が有効に機能するソリューションは存在しないと考えられる．また，医療情報ネットワークでは，端末側のシステムを変更するというアプローチをとることが困難である．電子カルテシステムは，単独のベンダがすべての機能を網羅的に提供しているのではなく，多くのベンダが協調して諸機能を提供するマルチベンダ型のシステムであるからである．電子カルテを閲覧するための端末には，各ベンダから提供された多くのアプリケーションが相互干渉しないように検証されたうえでインストールされている．また，各ベンダから提供される固有の端末も多く存在する．たとえば，これらの端末すべてに対して一律の構成変更を行う場合には，設定・検証作業のための多くの時間と費用が必要となる．さらに，これらの端末の中には，薬事法で定められた医療機器として承認されているものが存在する．医療機器承認番号を取得しているこれらの機器は，承認申請時の機器構成が原則として変更できない．

3. MediFlow：OpenFlow 医療情報ネットワーク

2 章での議論をもとに，OpenFlow を用いた医療情報ネットワーク「MediFlow」の設計と実装を行った．

OpenFlow の実装では，コントローラの制御アプリケーションが必要である．アプリケーション構築のためのさまざまなフレームワークが存在する．代表的なものに，NOX [11]，Beacon [12]，Maestro [13]，Floodlight [14]，Trema [15] 等があげられる．本研究では，OpenFlow の対応バージョンの多さと簡便さから Python で構築されている Ryu [16] を使用した．

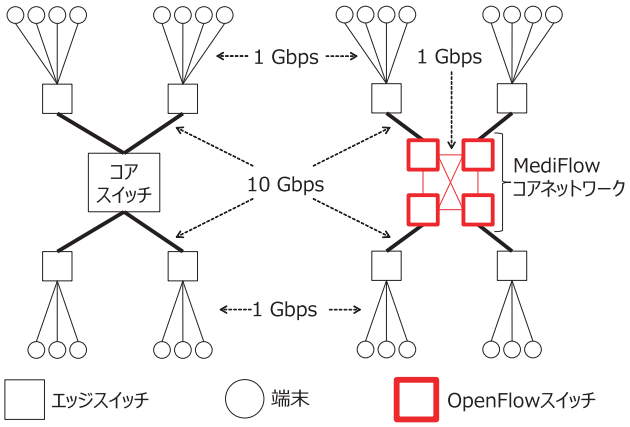


図 2 現在の医療情報ネットワーク（左）と MediFlow（右）
Fig. 2 Current medical information network (left) and MediFlow (right).

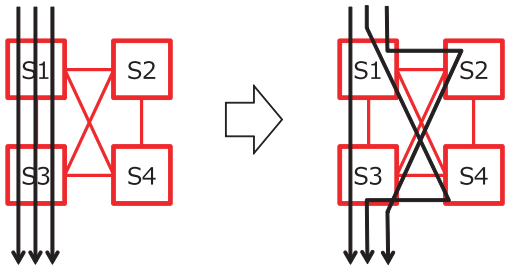


図 3 輻輳発生時の動作
Fig. 3 Operation of MediFlow when congestion is detected.

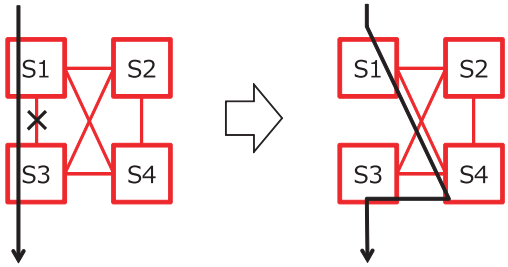


図 4 障害発生時の動作
Fig. 4 Operation of MediFlow when link is lost.

3.1 全体像

図 2 に本研究の基本的なアイディアを示す．MediFlow では，図 2 左のように高価格化の要因であった従来の医療情報ネットワークにおけるコアスイッチを，図 2 右のように MediFlow コアネットワークに置き換える．MediFlow コアネットワークとは，安価な OpenFlow スイッチをフルコネクトして構築したネットワークである．

MediFlow では，輻輳しているリンクを回避しながらフローを動的に経路に対して割り当てることで，各フローのスループットを向上させる．**図 3** に輻輳が発生した場合の MediFlow コアネットワークの動作を示す．OpenFlow スイッチ S1〜S4 は相互に 1 Gbps のリンクで接続されている．図 3 左のように，S1 から S3 に流れる 1 Gbps のフローが 3 つあった場合，S1 から S3 のリンクが輻輳を起こ

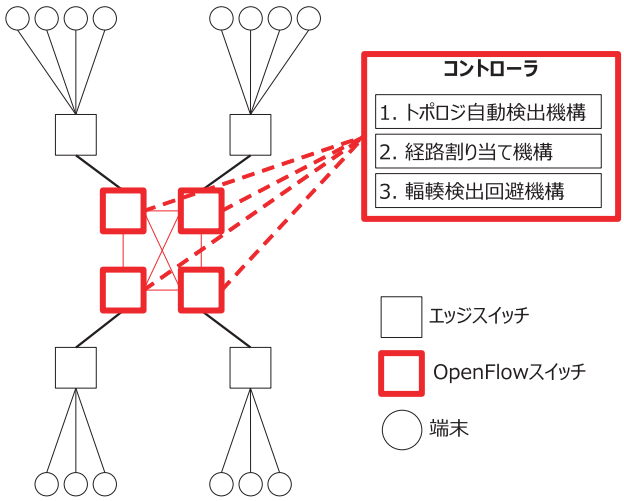


図 5 MediFlow の全体像
Fig. 5 Overview of MediFlow.

表 2 MediFlow のデータベース
Table 2 Database in MediFlow.

ofswitchTable	OpenFlow スイッチ一覧
linkTable	OpenFlow スイッチどうしの対応関係
hostTable	端末と OpenFlow スイッチの対応関係
pathTable	MediFlow コアネットワークのすべての経路
hopTable	各経路を構成するホップ
flowTable	フローと経路の対応関係
trafficTable	MediFlow コアネットワークのトラヒック量

して各フローは約 0.33 Gbps しかスループットがでなくなる．MediFlow では，図 3 右のようにフローに対して輻輳を避けるように動的に経路を割り当てることで，総スループットを向上させることができる．

また，MediFlow は障害発生時には障害が発生したリンクを避けるように動的に経路を割り当てることもできる．**図 4** に障害発生時の動作を示す．図 4 では，S1 から S3 に対して 1 Gbps の 1 つのフローが存在する．S1 から直接 S3 に配送していた際に S1 と S3 を接続するリンクに障害が発生した場合，自動的に S1→S4→S3 と経由するように経路を切り替えることができる．

図 5 に MediFlow の全体像を示す．各 OpenFlow スイッチはコントローラと接続されている．コントローラは，OpenFlow の機能で実現する
(1) トポロジ自動検出機構
(2) 経路割当て機構
(3) 輻輳検出回避機構
の 3 つの要素から構成される．すべての要素は OpenFlow の機能のみで実現可能であるため，端末側の変更は不要である．

表 2 にコントローラが具備するテーブルの一覧を示す．各テーブルの実装は MySQL 14.14 で行った．

ofswitchTable は MediFlow コアネットワークを構成する OpenFlow スイッチの情報を保持しているテーブルである．各 OpenFlow スイッチは datapath ID で一意に識別することができる．

linkTable は OpenFlow スイッチどうしの接続を保持しているテーブルである．接続の両端の OpenFlow スイッチの datapath ID，それぞれ接続しているポート番号を組として保持している．

pathTable は MediFlow コアネットワークが持ちうるすべての経路を保持したテーブルである．各項目は経路を一意に識別する path ID，経路の先頭の OpenFlow スイッチ，経路の終端の OpenFlow スイッチ，経路の長さを保持している．

hopTable は pathTable に保持されている経路がどの OpenFlow スイッチを経由して構成されるかの情報を保持している．各項目は path ID，datapath ID，ポート番号で構成される．MediFlow では，経路を経由する datapath ID と出力ポート番号の組の配列として表現している．ある path ID の経路でパケットを転送する場合に，ある datapath ID を持つ OpenFlow スイッチがどのポートで出力すればよいかの情報を各項目として保持している．

flowTable はフローと経路の対応関係を保持したテーブルである．本稿の説明では，説明の簡単化のためにフローを送信元 MAC アドレスと宛先 MAC アドレスで表現している．MediFlow はフローの識別に IP アドレスや TCP のポート番号を用いるように拡張可能である点に注意されたい．

trafficTable は OpenFlow スイッチ間を接続するリンクのトラフィック量を記録しているテーブルである．各項目はタイムスタンプ，datapath ID，ポート番号，累計送信データ量から構成されている．

3.2 トポロジ自動検出機構

トポロジ自動検出機構は MediFlow コアネットワークのトポロジを把握するための機構である．MediFlow では，OpenFlow の機能を利用して自動的にトポロジを把握することができるため，新しい OpenFlow スイッチを接続するだけで MediFlow コアネットワークを容易に拡張することができる．トポロジ自動検出機構で取得したトポロジ情報をもとに経路の割当てや輻輳を回避する経路を決定する．経路を算出するためには，
(1) OpenFlow スイッチどうしの相互接続状況
(2) 端末の接続状況
(3) リンクの切断
を把握する必要がある．

(1) OpenFlow スイッチどうしの相互接続状況は，OpenFlow の features メッセージと Link Layer Discovery Protocol（LLDP）パケットで把握する．MediFlow では，OpenFlow スイッチが接続されるときに発行される hello メッ

セージを受け取ると，features メッセージを用いて接続された OpenFlow スイッチの情報を取得する．features メッセージでは，OpenFlow スイッチ自体の MAC アドレスと，その OpenFlow スイッチの各ポート番号を取得することができる．取得した情報をもとに，MediFlow は各ポートから LLDP パケットを送信するように通知する．OpenFlow スイッチが LLDP パケットを受け取ると packet-in メッセージが発行される．packet-in メッセージにはどの OpenFlow スイッチからどのポートを経由して LLDP パケットを受け取ったかの情報が含まれている．

取得できる各 OpenFlow スイッチの MAC アドレスと LLDP パケットで取得できる情報を組み合わせることで OpenFlow スイッチどうしがどのように接続されているかを把握することができる．features メッセージで取得した情報は ofswitchTable に，LLDP パケットで取得した情報は linkTable に記録される．筆者らの実装では，Ryu のイベント `event.EventSwitchEnter`，`ofp_event.EventOFPStateChange`，`event.EventLinkAdd` に関連付けたハンドラで実装した．

(2) 端末の接続状況は，OpenFlow の packet-in メッセージを使用して把握する．packet-in メッセージとは，パケットが OpenFlow スイッチに届いた場合に，フローテーブルにそのパケットと適合するエントリが存在しなかった場合にコントローラに対して発行されるメッセージである．packet-in メッセージがコントローラに届いた際に，packet-in メッセージを送信した OpenFlow スイッチの識別子とポート番号，packet-in メッセージに含まれる送信元 MAC アドレスを取得する．送信元 MAC アドレスがコントローラに初めて届いた MAC アドレスであった場合に，取得した OpenFlow スイッチ識別子とポート番号の方向に端末が存在することを hostTable に記録する．筆者らの実装では，Ryu が持つイベント `event.EventHostAdd` に関連付けたハンドラに実装した．

(3) リンクの切断は，OpenFlow スイッチからの port-status メッセージと各 OpenFlow スイッチ間で定期的に LLDP パケットを交換することで検出する．port-status メッセージはポートの状態が変化したときに発行されるメッセージである．各 OpenFlow スイッチからの LLDP パケットの送信はコントローラから制御する．LLDP パケットに該当するフローエントリは作成しないように設計しているため，各 OpenFlow スイッチが LLDP パケットを受け取るとコントローラに必ず packet-in メッセージが送られる．コントローラにおいて，port-status メッセージでリンクダウンを受け取るか，各ポートから LLDP パケットを一定期間受け取らなかったと判定した場合には，そのリンクはダウンしたとしてトポロジ情報に反映する．筆者らの実装では，port-status メッセージによるリンクダウンや LLDP パケットの送受信は Ryu 内に隠蔽されているため，

Algorithm 1 packet-in イベントハンドラ

```
1: Input m: packet-in message
2: d_now ← getDatapathId(m)
3: a_src ← getSourceMacAddress(m)
4: a_dst ← getDestinationMacAddress(m)
5: p ← flowTable[(a_src, a_dst)]
6: if p = NULL then
7:     d_src ← hostTable[a_src]
8:     d_dst ← hostTalbe[a_dst]
9:     p ← getShortestFromPathTable(d_src, d_dst)
10:    flowTable[(a_src, a_dst)] ← p
11: end if
12: q ← getPortFromHopTable(d_now, p)
13: addFlowEntry(d_now, q, m)
14: sendPacketOut(d_now, q, m)
```

Ryu が持つイベント `ofp_event.EventOFPPortStatus` に関連付けたハンドラで実装した．

3.3 経路割当て機構

経路割当て機構は MediFlow コアネットワークにおいて端末間のパケットを転送するための経路を決定するための機構である．経路の決定のための計算コストが大きいと通信性能を損なってしまうため，少ない計算コストで実現できることが望ましい．たとえば，packet-in メッセージが来たタイミングで経路探索を行うとフローの種類の数だけ経路探索コストが生じてしまう．経路割当てにともなう計算コストを少なくすることを目的として MediFlow における経路割当て機構では

- 経路と端末を分離
- OpenFlow スイッチ間の経路のみ事前計算ですべて列挙

の 2 つの工夫を行っている．

経路と端末の分離は hostTable，pathTable，flowTable によって実現される．hostTable は端末の接続先情報を MAC アドレスから datapath ID に変換する仕組みである．flowTable はフローと経路を対応付けする仕組みである．MediFlow では hostTable と flowTable の存在により経路を datapath ID とポート番号だけで表現できるため，計算コストも OpenFlow スイッチの数だけ考慮すればよい．

また，経路は事前計算によってあらかじめすべて列挙されて pathTable と hopTable に記録されている．3.2 節で述べたトポロジ自動検出機構で取得した情報をもとに，各 OpenFlow スイッチ間のパスを幅優先探索で経路を発見する．MediFlow で用いる OpenFlow スイッチは多くても 10 台程度を想定しているため，幅優先探索で経路をすべて列挙しても計算量は問題にならない．3.3 節で述べる経路割当ての際に最短経路を選択できるように，各経路はホップ数と一緒に記録している．

Algorithm 1 に packet-in イベントハンドラのアルゴリズムを示す．コントローラに packet-in メッセージが到着

すると，packet-in メッセージに含まれる送信元 MAC アドレスと宛先 MAC アドレスを抽出する．抽出した送信元 MAC アドレスと宛先 MAC アドレスの組をフローとして，すでにその MAC アドレスの組に経路が割り当てられているかを調べる．もしすでに経路が割り当てられていた場合には次にどの OpenFlow スイッチに送るべきかを取得してフローエントリに登録するとともにパケットを送信する．

もしまだ経路が割り当てられていなかった場合には，抽出した宛先 MAC アドレスから hostTable の情報をもとに最終的にどの OpenFlow スイッチに送るべきかを取得する．次に，packet-in メッセージを受け取った OpenFlow スイッチの datapath ID を先頭，取得した OpenFlow スイッチの datapath ID を終端とする経路の中で最短の経路を pathTable から取得する．最後に，取得した経路をもとに次にどの OpenFlow スイッチに送るべきかの情報を取得してフローエントリと flowTable に登録するとともにパケットを送信する．

OpenFlow スイッチやリンクに障害が発生された場合には，まず，関連する情報を ofswitchTable，linkTable，pathTable，hopTable から削除する．通常であれば，次に行うべきは OpenFlow スイッチから関連するフローエントリを削除することである．しかしながら，OpenFlow スイッチから関連するフローエントリを消すと再度 packet-in メッセージが発生してオーバヘッドになってしまう．オーバヘッドを削減するために，MediFlow では，代替ルートを先に OpenFlow スイッチにフローエントリとして登録してから OpenFlow スイッチから障害が発生したリンクに関連するフローエントリを削除する．

3.4 輻輳検出回避機構

輻輳検出回避機構は，MediFlow コアネットワーク内のリンクで発生している輻輳を検出し，輻輳が発生しているリンクを回避するように経路を再設定する機構である．フローの数を F，ノード数を N とすると，フローに対して割当て可能な経路のバリエーションは $O(N^F)$ となる．医療情報ネットワークではフローは数が膨大かつ時々刻々と変化するため，可能な限り軽量な仕組みで輻輳の検出と回避ができることが望ましい．

MediFlow では，輻輳の検出を OpenFlow の機能を用いて各ポートの時間あたりの通信量を取得して閾値基準で輻輳の判定を行う．まず，コントローラは各 OpenFlow スイッチに対して multipart メッセージを用いて port statistics request を定期的に発行する．このとき，OpenFlow スイッチどうしを接続しているポートのみの情報を要求することでトラヒック情報を取得する際の負荷を削減している．各 OpenFlow スイッチから port statistics reply が返ってくると port statistics reply イベントハンドラを実行する．

Algorithm 2 に port statistics reply イベントハンドラの動作を示す．port statistics reply が返ってきたらそのメッセージからそのメッセージを送った OpenFlow スイッチの datapath ID，ポート番号，送信バイト数を取得する．OpenFlow の port statistics reply に含まれている送信バイト数はその時点までの累計値である．現段階でのそのポートのトラヒック量を算出するために，その datapath ID，ポート番号において前回取得した送信バイト数 b_{pre} と取得した時刻 t_{pre} を getPreviousTrafficRecord 関数を用いて取得する．最新の送信バイト数と前回取得した送信バイト数の差分，現在の時刻と前回取得した時刻の差分から現在のそのポートにおけるトラヒック量を算出する．トラヒック量が閾値 α を超えていた場合には輻輳が発生していると判定して congested_event_handler 関数を実行する．congested_event_handler 関数に関しては後述する．閾値 α は現在の実装ではワイヤレートの 90%を使用している．たとえばワイヤレートが 1 Gbps のリンクの場合には，α は 900 Mbps となる．最後に取得した送信バイト数を insertTrafficStats 関数でデータベースに記録する．

Algorithm 2 port statistics reply イベントハンドラ

```
1: Input m: port statistics reply message
2: d ← getDatapathId(m)
3: p ← getPortNumber(m)
4: b ← getTxBytes(m)
5: t ← now()
6: (t_pre, b_pre) ← getPreviousFromTrafficTable(d, p)
7: r ← (b − b_pre)/(t − t_pre)
8: if r > α then
9:     call congested_event_handler(d, p)
10: end if
11: insertToTrafficTable(d, p, t, b)
```

congested_event_handler 関数は輻輳しているリンクを回避するように経路を再割当てする関数である．関数内では経路の計算は行わず，3.3 節で述べたあらかじめ算出済みの経路をランダムに選択するだけの処理で実現している．ランダム性を導入しているのも軽量な処理で輻輳を回避するためである．ランダムに経路を選択する処理はほとんど計算コストがかからない．経路選択後に再び輻輳が発生する場合には，再度 congested_event_handler 関数が呼ばれるため，最終的に適切な経路割当てが行われるか，輻輳を起こすトラヒックがなくなるかするまで経路の再割当てが行われ続ける．

Algorithm 3 に congested_event_handler 関数の動作を示す．congested_event_handler 関数は入力としてどの datapath ID のどのポートで輻輳が起こっているかの情報が与えられる．まず，引数として与えられた datapath ID d とポート番号 p を用いて，すでに経路の割り当てられたフローの中で (d, p) を使用しているフローをすべて取得する．もし取得したフローが 1 つしかなかった場合，経路の変更

Algorithm 3 congested_event_handler 関数

```
1: Input d: datapath ID, p: port number
2: F ← getAllocatedFlowSet(d, p)
3: if |F| = 1 then
4:     return
5: end if
6: (a_src, a_dst) ← choose a flow randomly from F
7: Q ← getPathSetFromPathTable(a_src, a_dst)
8: while |Q| > 0 do
9:     q_new ← get a path randomly from Q
10:    if q_new does not include (d, p) then
11:        q_old ← flowTable[(a_src, a_dst)]
12:        flowTable[(a_src, a_dst)] ← q_new
13:        add flow entries related to q_new
14:        delete flow entries related to q_old
15:        return
16:    end if
17:    Q = Q − q
18: end while
```

は必要ないので関数の処理を終了する．もし (d, p) を含む経路に割り当てられたフローが 2 個以上存在した場合には，その中から 1 つを経路変更候補としてランダムに選択する．7 行目では，経路変更候補のフローに割当て可能な経路をすべて取得する．8 行目から 17 行目では，集合 Q の中から 1 つずつ経路を取り出し，現在輻輳が起きているリンクを含まない場合にはその経路をフロー $(a_{\mathrm{src}}, a_{\mathrm{dst}})$ に割り当てる新しい経路として利用する．

3.2 節で述べた障害発生時と同様に，経路変更時のオーバヘッドを削減する仕組みを輻輳発生時にも導入している．具体的には，Algorithm 3 の 13 行目において先に新しい経路のフローエントリを関連する OpenFlow スイッチに登録してから 14 行目において古い経路のフローエントリを削除することでシームレスな経路変更を実現している．

4. 実機による評価

3 章に示した MediFlow において，基本性能を検証することを目的として実機による評価を行った．

4.1 評価環境

図 6 に実機評価環境を示す．3 台の OpenFlow スイッチ（S1〜S3），1 台の L2 スイッチ，1 台のコントローラ，1 台のトラヒックジェネレータと 2 台のパケットキャプチャ用 PC（PC1，PC2）から構成される．

S1〜S3 の 3 台の OpenFlow スイッチは 1 Gbps のリンクで相互に接続されている．OpenFlow スイッチには，48 ポートの 1000Base-T インタフェースを具備した PICA8 P-3297 [17] を用いた．PICA8 P-3297 では，L3 スイッチとして動作する専用 OS（PicOS$^{\mathrm{TM}}$）と Linux のデュアルブートが可能である．Linux モードでは Open vSwitch での動作が可能である．OpenFlow バージョン 1.0 から 1.4 までの各プロトコルに対応している．コントローラから送信

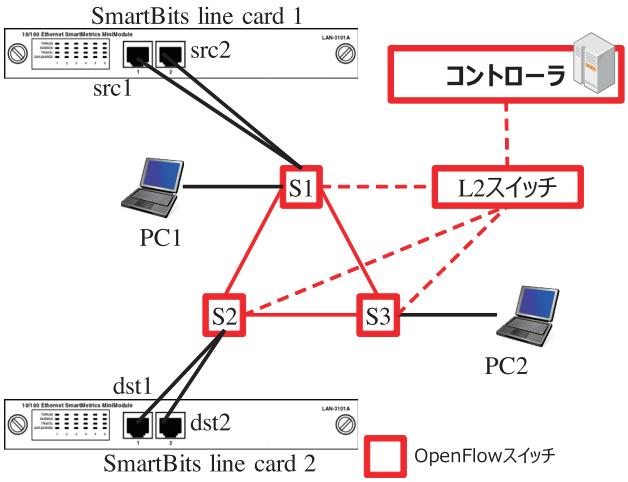


図 6 実機実験環境

Fig. 6 Experimental environment with actual machines.

されるフローエントリは OpenFlow スイッチ上の Ternary Content Addressable Memory（TCAM）にストアされる．コントローラは L2 スイッチを介して各 OpenFlow スイッチと接続されている．

コントローラには HP Pavilion HPE h9（CPU: Corei7 3770 3.4 GHz，RAM: 16 GB，HDD: 2 TB × 3）を用いた．この PC 上では Ubuntu 16.04 が動作している．コントローラの実装には Ryu 4.9 を用いた．L2 スイッチには HP ProCurve Switch 2610-48-PWR を用いた

パケットキャプチャ用の PC1 と PC2 にはそれぞれ Microsoft Surface Pro3（CPU: Corei7-4650U 2.5 GHz，RAM: 8 GB，SSD: 256 GB）と Microsoft Surface Pro4（CPU: Corei5-6300U 2.5 GHz，RAM: 8 GB，SSD: 256 GB）を用いた．PC1 は S1 から S2 のリンクを，PC2 は S3 から S2 へのリンクをそれぞれミラーポートを介してモニタリングしている．

トラヒックジェネレータには，Spirent Communications 社の SmartBits 600B [18] を用いた．SmartBits は 2 枚のラインカードを具備している．それぞれのラインカードは 1000Base-T のイーサネットポートを 2 つずつ（src1，src2，dst1，dst2）有している．各 OpenFlow スイッチとは 1 Gbps のリンクで接続した．SmartBits は任意のフレーム長，任意のプロトコルで任意の流量を生成することができる．評価では，フレーム長は 1,518 B，プロトコルは TCP とした．

4.2 価格

現在の医療情報ネットワークで用いているコアスイッチは価格が数千万円である．たとえば，600 床の病院において約 6,000 万円のコアスイッチを導入している例がある．それに対して OpenFlow スイッチは 1 台あたり数十万円である．本稿の実機評価で用いている PICA8 P-3297 は 2017 年 3 月時点で約 35 万円で購入することができる．仮に 10

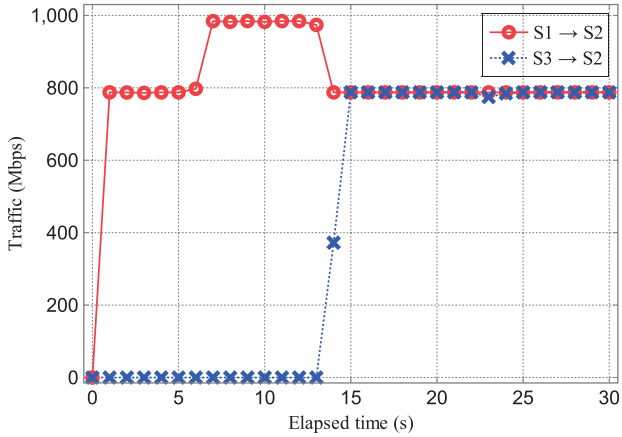


図 7 輻輳時のトラヒック推移
Fig. 7 Traffic trend on congestion.

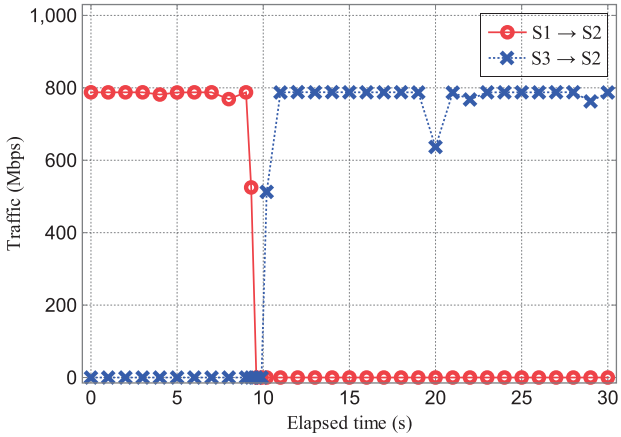


図 8 リンクロスト時のトラヒック推移
Fig. 8 Traffic trend on link lost.

台の OpenFlow スイッチを用いたとしても価格は約 350 万円程度であり，費用を大幅に削減できていることが分かる．

4.3 スループット

3 章に示した MediFlow におけるフローに対する経路割当てでスループットを向上させることができることを確認するために，複数フローを発生させた場合のスループットの評価を行った．具体的には，図 6 の実験装置において，実験開始直後に src1 から dst1 への 800 Mbps のトラヒックを，実験開始から 6 秒後に src2 から dst2 への 800 Mbps のトラヒックを発生させた．輻輳検出に用いる閾値（Algorithm 2 中の α）はワイヤレートの 90%である 900 Mbps とした．

図 7 に評価結果を示す．横軸が実験開始からの経過時間，縦軸がトラヒックである．S1→S2 のリンクと S3→S2 のリンクをプロットしている．実験開始直後は，src1 から dst1 への 800 Mbps のトラヒックのみが S1→S2 のリンクで観測できている．実験開始から 6 秒後に src2 から dst2 への 800 Mbps のトラヒックが発生すると，S1→S2 のリンクに与えられるトラヒックが合計で 1600 Mbps となり，輻輳が発生していることが分かる．実験開始から 14 秒後に，輻輳検出回避機構によって src2 から dst2 のフローに対して S1→S3→S2 の経路が割り当てられ，S1→S2 のリンクが 800 Mbps に，S3→S2 のリンクも 800 Mbps になっていることが分かる．

輻輳の検出と輻輳の回避に約 7 秒を要しているのは今回の MediFlow の実装依存である．医療情報ネットワークにおけるバックアップのように数十分から数時間継続するフローに対しては 7 秒の切替え時間でも運用上は問題は生じない．約 7 秒の遅れは各 OpenFlow スイッチに 5 秒間隔でトラヒック情報の問合せを行っていることに起因している．OpenFlow スイッチへのトラヒック情報の問合せ間隔を短くすることで輻輳回避の時間に要する時間を数秒程度まで短くすることができる．切替え時間の下限は Management Information Base（MIB）の実装に依存する．今回の実装に用いている PICA8 P-3297 では，OpenFlow で取得できるトラヒック情報は SNMP で利用している MIB を用いて実装されているからである．

4.4 ロバスト性

医療情報ネットワークにおけるネットワーク障害は大別すると 2 つのパターンに分類できる．1 つ目はノードそのものの故障，2 つ目はノード内の特定のポートの故障である．発生頻度が高い後者の障害が発生した場合の挙動を確認することを目的としてロバスト性の評価を行った．具体的には，src1 から dst1 に対して 800 Mbps のトラヒックを発生させた 9 秒後にポートの障害として S1→S3 に接続されているケーブルを抜去した．

図 8 に評価結果を示す．横軸が実験開始からの経過時間，縦軸がトラヒックである．4.3 節の評価と同様に，S1→S2 のリンクと S3→S2 のリンクのトラヒックをプロットしている．実験開始直後は src1 から dst1 のフローに対して S1→S2 の経路が割り当てられるため，図 8 に S1→S2 のトラヒックが観測できている．実験開始から 9 秒後に S1 と S2 を接続するケーブルを抜去すると，S1→S2 のリンクのトラヒックはゼロとなる．S1 と S2 を接続するケーブルを抜去してから約 0.7 秒後に，MediFlow が src1 から dst1 のフローに対して S1→S3→S2 の経路を割り当てるため，図 8 でも S3→S2 のリンクのトラヒックが発生している．

障害の発生から経路の再構築まで約 0.7 秒を要しているのは OpenFlow スイッチの実装依存であると考えている．OpenFlow スイッチでは，ポートの状態が変化したことを port-status メッセージで通知している．MediFlow では，port-status メッセージの到着に合わせて新しい経路のフローエントリの追加と削除を行っている．リンク断の検出，通知，フローエントリの登録，フローエントリの削除それぞれに要する時間を合算すると約 0.7 秒になるのだと考えている．Microsoft Windows における TCP セション

タイムアウトのデフォルト値は 21 秒，Linux は 189 秒であることから，約 0.7 秒の間リンクがロストしていても実運用上の問題は生じない．

5. シミュレーションによる評価

4 章の評価では実機を用いてミクロな観点から MediFlow の評価を行った．それに対して本章では，計算機シミュレーションを用いてマクロな観点から MediFlow の評価を行う．

5.1 評価環境

図 9 に本評価で用いる実際の医療情報ネットワークを基準にしたトポロジを示す．各ネットワークは OpenFlow スイッチ，エッジスイッチ，端末で構成される．OpenFlow スイッチの数は OpenFlow スイッチの数が増えた場合にスケールアウトできているかどうかを確認するために 4 台から 10 台と変化させた．各 OpenFlow スイッチは相互に各 1 Gbps でフルコネクトで接続して MediFlow コアネットワークを構築している．エッジスイッチの数は実際の医療情報ネットワークと揃えることを目的として 4 台の固定とした．端末は各フローごとに送信元と宛先が 1 台ずつ独立に存在する．各フローは 1 Gbps のトラヒックとした．本評価のもととなった医療情報ネットワークではエッジスイッチと OpenFlow スイッチは 10 Gbps のリンクで接続されている．評価結果の分かりやすさを考慮して，本評価ではエッジスイッチと OpenFlow スイッチは無限の容量を持つリンクで接続されていることとした．比較対象として図 2 の左側（現在の医療情報ネットワーク）に示す現在の医療情報ネットワークシステムで用いられているコアスイッチ（グラフ中の core switch）を用いた．コアスイッチの性能は無限大としている．

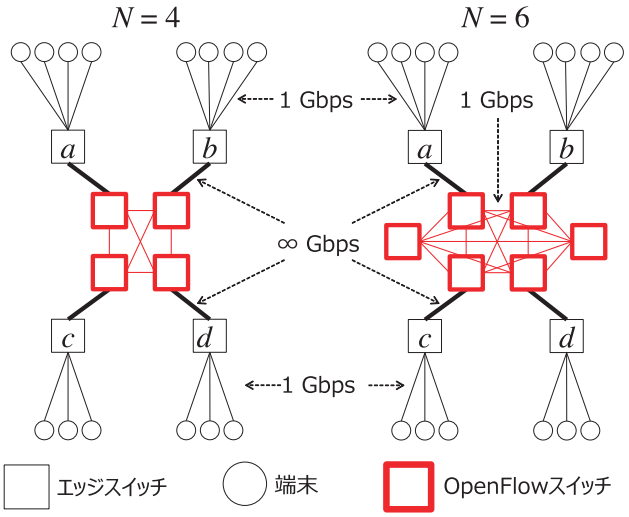


図 9 評価トポロジ．$N = 4$ の場合（左）と $N = 6$ の場合（右）
Fig. 9 Evaluation topology.

5.2 偏ったフローに対する評価

偏ったフローが発生した場合の MediFlow の性能を計算機シミュレーションによって評価した．偏ったフローとは，MediFlow コアネットワークに対してある時間帯に特定のエッジスイッチ方面からの流入と，ある特定のエッジスイッチ方面への流出が集中した状態を模擬している．図 9 におけるエッジスイッチ a からエッジスイッチ b へのフローが複数発生している状態である．偏ったフローの例としては，深夜帯に発生するバックアップタスクによって生じるフローがあげられる．

図 10 に偏ったフローでの評価結果を示す．横軸はエッジスイッチ a からエッジスイッチ b へのフロー数を，縦軸は端末間での総スループットを示している．MediFlow コアネットワークを構成する OpenFlow スイッチの台数 N を変化させた場合の各フロー数における総スループットと，既存のコアスイッチを用いた場合（core switch）をそれぞれプロットしている．既存のコアスイッチでは，与えられたフロー数に比例してスループットが増加していることが確認できる．一方で，MediFlow では OpenFlow スイッチの数に応じて上限が存在しているものの，OpenFlow スイッチの数を増やすと上限がスイッチ数に比例して増加していることが分かる．また，上限の増加は既存のコアスイッチに漸近していることも分かる．複数のフローが流入元のエッジスイッチと流出先のエッジスイッチを共有しているため，各フローを 1 Gbps とすると上限は $N - 1$ [Gbps] となっている．

5.3 ランダムなフローに対する評価

5.2 節の評価では，偏ったフローにおける評価であった．実際の医療情報ネットワークでは，多様なトラヒックが多様な流入元，流出先の組合せで発生する時間帯も存在すると考えられる．このような観点から，流入エッジスイッチと流出エッジスイッチをランダムに選択してフローを発生させた場合の評価を行った．

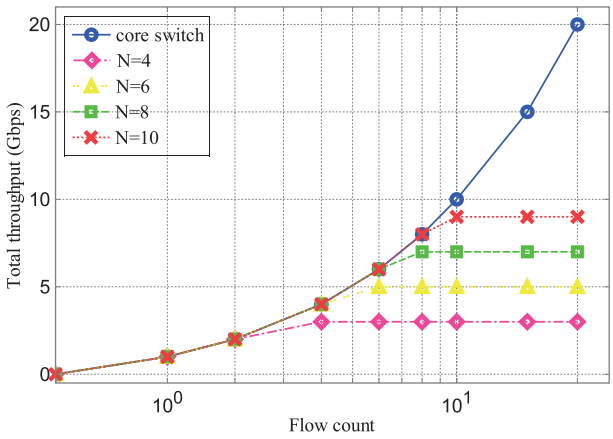


図 10 偏ったフローによる評価
Fig. 10 Evaluation at one-sided flows.

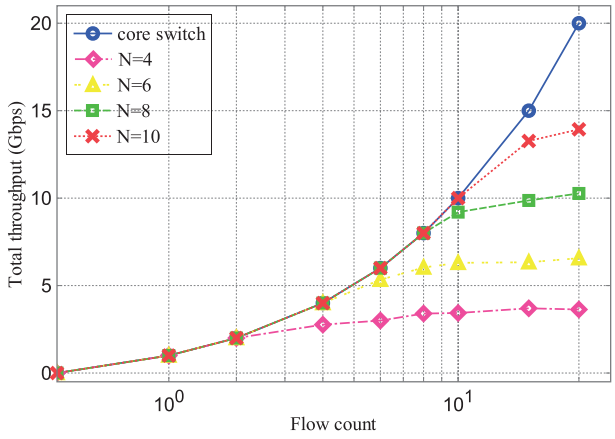


図 11 ランダムなフローによる評価
Fig. 11 Evaluation at random flows.

図 11 にランダムなフローにおける評価結果を示す．横軸はフロー数，縦軸はスループットを示している．5.2 節と同様に，MediFlow コアネットワークを構成するスイッチ台数 N が変化した場合の各フロー数におけるスループットをプロットしている．MediFlow では OpenFlow スイッチの数に応じて上限が存在しているものの，OpenFlow スイッチの数を増やすと上限がスイッチ数に比例して増加していることが分かる．また，上限の増加は既存のコアスイッチに漸近していることも分かる．さらに，各 OpenFlow スイッチ数での性能はそれぞれ 5.2 節の総スループットより大きいことも分かる．MediFlow の仕組みにより，流入元スイッチ，流出先スイッチが多様な場合にはフローが偏っている場合に比べてより多様なリンクの容量を有効活用できているからだと考えられる．

6. 関連研究

筆者らの知りうる限り，医療情報ネットワークのコアスイッチを複数の OpenFlow スイッチをフルコネクトしたネットワークに置き換えてシステムとして検証した先行研究は存在しない．

医療情報ネットワークへの SDN 導入事例としては，日本電気による金沢大学附属病院，慶應義塾大学病院，名古屋市立大学病院，京都岡本記念病院への導入があげられる．4 つの病院において共通していることは，ネットワークインフラに着目して複数の物理ネットワークを Virtual Tenant Network（VTN）を用いて仮想化を行うことで，管理コストの低減や利便性の向上を図っていることである．たとえば慶應義塾大学病院では，診療系ネットワークと研究系ネットワークが物理的にそれぞれ独立して構築されており，これらのネットワークを連携することによって利用者の利便性向上を図りたいという要件があった．また，医療系と研究系のそれぞれのネットワークに対して異なるセキュリティレベルを適用しなければならないという課題も存在していた．このような要件と課題を満たすために，診療系ネットワークと研究系ネットワークに対してそれぞれ VTN の割当てを行っている．各 VTN に個別のセキュリティレベルを適用することで，利便性と安全性を同時に実現している事例である．それに対して本研究は，利便性や安全性よりも，医療情報ネットワークにおける増大するスループットへの対処とロバスト性の確保を目的としたトラヒック制御に着目している研究であると位置付けられる．

MedFlow を構成する個々の技術に関連する研究としては，Software Defined Network（SDN）の研究があげられる．SDN の取り組みは，WAN（Wide Area Networks）向けのものと LAN（Local Area Networks）の向けのものに分けることができる．

WAN 向けの SDN としては，文献 [19], [20], [21], [22], [23], [24], [25], [26] があげられる．たとえば文献 [22] では，広域ネットワーク障害における復帰時間の短縮が取り組まれている．文献 [23] や文献 [25] では，SDN をネットワークセキュリティに適用する応用が検討されている．文献 [21] では，広帯域を必要とする高精細映像の伝送のために，動的な仮想マルチパス制御機能を提案している．WAN において動的に経路を制御してスループットを向上させる場合，本研究で対象としている医療情報ネットワーク等の LAN とは要求と課題が異なる．たとえば，WAN に OpenFlow を適用しようとした場合，LAN に比べてフローの種類が莫大であるため，OpenFlow でサポートできるフローエントリに登録できるフロー数の少なさをどのように克服するかが課題となる．また，管理者の異なる複数のネットワークにおけるトラヒックをどのように把握するかも課題となる．

LAN 向けの SDN としては，OpenFlow を用いているものと OpenFlow を用いていないものに分けられる．OpenFlow を用いていないものとしては，データセンタを対象とした文献 [9], [10], [27], [28] があげられる．データセンタでは，端末とネットワークをまとめて 1 つの体系で管理できるため，端末側のソフトウェアを変更してのアプローチが容易である．たとえば文献 [10] では，データセンタネットワークを構成する端末のトランスポートプロトコルを刷新することで性能向上を実現している．しかしながら，2 章で述べたとおり，本研究で対象としている医療情報ネットワークでは端末側のソフトウェアを変更することが困難である．

OpenFlow に類似した既存技術としては，イーサネットファブリックがあげられる．イーサネットファブリックでは，従来の階層型トポロジをフラット型トポロジに移行することによって，ネットワークの運用管理の簡素化と効率化を実現している．イーサネットファブリックは IETF 等で標準化が進められているものの [29]，対応している製品が少なく，現状ではイーサネットファブリックに対応したスイッチは高価である．たとえば，2017 年 8 月現在で Brocade 社が提供している VDX6740 は，最小構成時で

標準価格が 367 万円となっている．しかしながら，ファブリックのハードウェア構成は OpenFlow スイッチと同様に既存のボックス型スイッチと類似しているため，普及が進めば価格が下がる可能性がある．仮に普及が進み，OpenFlow における NOX，Trema，Ryu のような使いやすいツール群が揃った場合には，MediFlow と同様の仕組みを構築することが可能であると考えている．

LAN 向けの SDN の中で OpenFlow を用いているものでは，スループットの向上を対象としているものとそれ以外に分けることができる．スループットの向上以外を目的とした研究としては，文献 [30], [31], [32], [33] があげられる．たとえば文献 [32] では，OpenFlow を活用して，キャンパスネットワークにおける VLAN 管理コストの削減を目指している．文献 [30], [31] では，OpenFlow を用いたロードバランサを検討している．

LAN 向けの SDN の中で OpenFlow を用いているものでは，スループットの向上を対象としている研究が部分的には本研究と最も関連が強い．トラヒックの情報の取得方法や経路の選択方法の研究として，データセンタ向けのネットワーク等比較的端末側のソフトウェアの変更が可能な領域を対象として，端末装置から直接トラヒックの情報を取得したり端末側のミドルウェアを変更して端末装置の要求に応じて経路を構築したりする研究があげられる [34], [35], [36], [37], [38], [39], [40], [41], [42], [43].

たとえば，文献 [40] では，アプリケーションが直接コントローラに対してアプリケーションのトラヒック要求を通知する仕組みを前提としている．しかしながら，2 章で述べたように，本研究で対象としている医療情報ネットワークでは端末側のソフトウェアを変更することが困難であるのでこれらの手法は適用できない．

経路の決定方法として，各リンクのコストをモニタリング等で算出してリンクコストを考慮した経路決定アルゴリズムによってマルチパスルーティングや低遅延性を決定している手法が検討されている [44], [45], [46], [47], [48], [49]．たとえば，文献 [45] では，リンクコストに基づいて非集計ロジックモデルを用いたマルチパスルーティング制御方式を提案している．各リンクのコストを正確に見積もることができ，トラヒックの変動が少ないネットワークであればこれらの手法は有効であると考えられる．しかしながら，筆者らの実装で用いた OpenFlow スイッチでは，取得できるトラヒックの情報の更新間隔が遅く，実際の医療情報ネットワークではトラヒックの時変動性が大きいため，リンクコストを正確に見積もるのが困難である．特に発生しているトラヒックが定常的に発生しているものなのか，瞬間的に発生しているものなのかを区別するのは難しい．また，多様なフローが次々と発生してリンクコストが頻繁に変動しうる環境では，経路の再計算も高頻度に発生してパフォーマンスに影響することも考えられる．さらに，リンクコストのメトリックと経路選択アルゴリズムによっては解が存在しないという事態も発生しうる．このような観点から，本研究では，「あるリンクにおいて輻輳が発生した」という単純な情報のみをトリガとして，あらかじめ算出済みの経路をランダムに選択する軽量で単純なアプローチを採用している．

7. 議論

7.1 適用可能性に関する議論

コアスイッチ，エッジスイッチ，端末の 3 層構造は医療情報ネットワークだけでなく，エンタープライズネットワーク等でも採用されている構成である．すなわち，MediFlow における「コアスイッチをフルコネクトされた複数の OpenFlow スイッチに置き換える」という仕組み自体は医療情報ネットワークだけでなく，エンタープライズネットワークにも適用できる可能性がある．しかしながら，以下の 4 点に関して考慮する必要がある．

1 つめは，ネットワークのサイズである．MediFlow では OpenFlow の機能によってフローを識別しているが，ネットワークの規模が大きくなると packet-in メッセージが多発し，ネットワーク性能に影響が出る可能性がある．OpenFlow スイッチはパケットのフォワーディングルールを TCAM にエントリしている．一般に TCAM は複雑な回路を用いているがゆえに高速検索が可能である反面，小容量であることが問題となる．適用したネットワークのサイズとアプリケーションによってマッチングルールが膨大となった場合には，TCAM が溢れ，packet-in メッセージが頻発することになる．サイズの大きなネットワークに適用する場合には，フォワーディングルールの統合やタイムアウト値の自動調整等，限られた TCAM を有効活用するための研究開発 [50] が必要であると考えている．

2 つめは，障害時の復帰時間である．MediFlow の実装では，リンクロストから約 0.7 秒での復帰を実現した．現在の医療情報ネットワークで利用しているアプリケーションでは 0.7 秒のリンクロストは問題にならない．しかしながら，遠隔手術等リアルタイムの映像と音声を扱うようなサービスの場合には 0.7 秒のリンクロストが問題になる可能性がある．たとえば，ITU における IP 電話の品質に関する推奨 G.114 [51] が要求する音声アプリケーションのネットワーク遅延は 0.4 秒未満となっている．リンクロストを高速に検出するための仕組みの研究開発が必要であると考えている．

3 つめは，コントローラである．今回の論文では，主にスイッチに焦点を当てていたため，コントローラに関するさらなる検討が必要であると認識している．たとえば前述したようにネットワークのサイズが大きくなると packet-in が多発すると考えられる．また，コントローラの障害に関しても検討が必要となる．たとえば，文献 [52] で示されて

いるように，複数の PC でコントローラを構成するための冗長化手法に関する研究開発が必要であると考えている．

4 つめは移動端末への対応である．2.2 節で述べたとおり，医療情報ネットワークでは移動端末が多く用いられている．端末が移動した場合には，接続先の OpenFlow スイッチが変わることが考えられる．現状の MediFlow では，端末への経路を接続しているポートを起点として決定しているため，端末と OpenFlow スイッチの接続関係が変わった場合には動作しない可能性がある．端末の移動に対応するためには，hostTable と flowTable を更新するとともに古いフローエントリを削除するための仕組みを新たに導入する必要がある．

たとえば，以下のような拡張が考えられる．まず，MediFlow コアネットワークにおいて各 OpenFlow スイッチのポートが OpenFlow スイッチどうしを接続しているポートなのか，MediFlow コアネットワークの外と接続しているポートなのかを区別する．端末が移動してフローエントリに登録されていない経路を利用する場合には packet-in メッセージが発生する．packet-in メッセージが発生しているタイミングで端末が接続されているポートを確認し，過去に接続されていたポートと異なる場合には hostTable と flowTable を更新するとともに，関連するフローエントリをすべて削除する．

ただし，医療情報ネットワークの構造と病院の運用形態によっては問題にならない場合もある．多くの病院では，病棟内や外来診療における診療業務は各看護単位，もしくは診療科単位に行われている．たとえば，病棟内で建屋の各階ごとにに看護単位が決められている場合には，病棟内に設置されたすべてのエッジスイッチを同一の OpenFlow スイッチ配下に接続すればよい．端末が移動しても端末と OpenFlow スイッチの関係は変わらない．

7.2 スイッチの価格差要因に関する議論

4.2 節で示しているとおり，MediFlow は既存のコアスイッチと比較して低コストで導入可能である．本節では低価格化の要因に関して議論する．

コアスイッチの高価格化要因について，提案手法で利用しているボックス型スイッチ PICA8 P-3297 と，比較対象として想定しているコアスイッチ CISCO Catalyst4500 の主要諸元に関する比較を行った．**表 3** に比較結果を示す．表 3 では，単位時間あたりの最大データ処理量（throughput），単位時間当たりの最大パケット転送量（IPv4 forward），スイッチが有するバッファメモリの量（buffer RAM），MAC アドレスの最大登録数（MAC entry），VLAN 最大登録数（VLAN），スイッチの主記憶容量（RAM），スイッチの CPU コア数と最大稼働周波数（CPU）を比較している．品質面においては，各機器の可用性として Mean Time Between Failure（MTBF）の比較を行った．コアスイッチは，シャシー，スイッチングモジュール，電源等の複数の要素から構成されているため，この要素中で最もボックス型スイッチに類似したスイッチングモジュールの値を掲載した．

表 3 コアスイッチと OpenFlow スイッチの主要諸元
Table 3 Specification of core switch vs. OpenFlow switch.

	Catalyst4500	PICA8 P-3297
throughput	800 Gbps	176 Gbps
IPv4 forward	250 Mpps	132 Mpps
buffer RAM	32 MB	4 MB
MAC entry	55 K	32 K
VLAN	4,094	4,094
RAM	4 GB	2 GB
CPU	Dual Core 1.5 GHz	Dual Core 1.33 GHz
MTBF	209,330h	196,356h

表 3 に示した比較項目において，ボックス型スイッチに対するコアスイッチの性能比は throughput では約 4.5 倍，IPv4 forward では約 1.9 倍，buffer RAM では 8 倍，MAC entry では 1.7 倍，VLAN では同値，RAM では 2 倍，CPU では約 1.1 倍を示している．また，品質を示す MTBF は約 1.06 倍となっている．一方価格比では 4.2 節で述べたとおりおおむね 171 倍の価格差がある．

コアスイッチとボックス型スイッチの価格に大きな差が出てくるのは量産品であるかどうかの違いであると考えられる．コアスイッチはモジュール構成型であるものの，個々の構成パーツはベンダごとに互換性がないため，生産量が少なくなって開発コストが高くなっていると考えている．

生産量と開発コストの関係を示す法則として，経験曲線効果が知られている [53]．経験曲線効果で用いられる習熟率とは，ある製品を繰り返し生産するとき，その製品の単位生産工数が逓減する習熟現象を表している．この習熟率は産業分野によって異なるといわれている．

5.3 節において，コアスイッチに漸近する性能を示したボックス型スイッチ 10 台とコアスイッチの価格には約 17.1 倍の価格差が認められる．OpenFlow スイッチのハードウェア構成は一般に流通するボックス型スイッチと同等であり，ソフトウェアを入れ替えるだけで OpenFlow スイッチは実現できる．量産品であるボックス型スイッチの生産量と，各エンタープライズネットワークごとにカスタマイズ生産するコアスイッチの生産量が大きく異なることによって価格差が生まれていると考えている．たとえば，経験曲線効果によれば，習熟率 60%のときは生産数で約 47 倍，習熟率 70%のときは生産数で約 250 倍，習熟率 80%のときは生産数で約 6,800 倍の差がある場合に約 17.1 倍の価格差が生まれる．

8. おわりに

本稿では，現在の医療情報ネットワークの高価格化を

解決する手法として，OpenFlow を用いた医療情報ネットワーク MediFlow を提案した．MediFlow では，従来の医療情報ネットワークの高価なコアスイッチを MediFlow コアネットワークに置き換えることで高価格化の問題を解決する．MediFlow コアネットワークは，フルコネクトされた複数の安価な OpenFlow スイッチで構成される．フローに応じて動的に経路を変更することで，時々刻々と変化するトラヒックの変動やリンク断等の障害にも対応できる．MediFlow を市販されている OpenFlow スイッチ上に実装し，トラヒックの変動や障害に対応できることを実験によって確認できた．複数台のノードで構成される MediFlow コアネットワークを用いたシミュレーションでは，MediFlow が現在の医療情報ネットワークで利用されているコアスイッチに漸近する性能を有することが確認できた．

謝辞 本研究は JSPS 科研費 JP16H01718 等の助成を受けて行った．

参考文献

[1] 厚生労働省：保健医療分野の情報化にむけてのグランドデザインの策定について，入手先 ⟨http://www.mhlw.go.jp/shingi/0112/s1226-1.html⟩.

[2] 一般社団法人保健医療福祉情報システム工業会：オーダリング・電子カルテシステム病院導入状況調査報告書，入手先 ⟨https://www.jahis.jp⟩.

[3] Mckeown, N., Shenker, S., Anderson, T., Peterson, L., Turner, J., Balakrishnan, H. and Rexford, J.: OpenFlow: Enabling Innovation in Campus Networks, *ACM SIGCOMM Computer Communication Review*, Vol.38, pp.69–74 (2008).

[4] Kolasani, A. and Ramamurthy, B.: Network Innovation using OpenFlow: A Survey, *IEEE Communications Surveys and Tutorials*, Vol.16, pp.493–512 (2014).

[5] Hu, F., Hao, Q. and Bao, K.: A Survey on Software-Defined Network and OpenFlow: From Concept to Implementation, *IEEE Communications Surveys and Tutorials*, Vol.16, pp.2181–2206 (2014).

[6] Fluke Networks: Network Time Machine, available from ⟨http://www.toyo.co.jp/ict/products/detail/clearsight.html⟩.

[7] 田中勝弥：SINET L2VPN を用いた国立大学病院災害対策医療情報システムにおける遠隔バックアップシステムの構築 (2014)，入手先 ⟨http://w4a.sinet.ad.jp/storage/advnet2014-04.pdf⟩.

[8] 荒木賢二，中武豊伸：IT System Innovation Review Android 対応システム「WATATUMI」のパフォーマンスを検証する，月刊医療，Vol.38, pp.60–62 (2011).

[9] Raiciu, C., Barre, S., Pluntke, C., Greenhalgh, A., Wischik, D. and H, M.: Improving Datacenter Performance and Robustness with Multipath TCP, *ACM SIGCOMM Computer Communication Review*, Vol.41, pp.266–277 (2011).

[10] Cao, Y., Xu, M., Fu, X. and Dong, E.: Explicit Multipath Congestion Control for Data Center Networks, *Proc. ACM 9th Conference on Emerging Networking Experiments and Technologies* (*ACM CoNEXT'13*), pp.73–84 (2013).

[11] Gude, N., Koponen, T., Pettit, J., Pfaff, B., Casado, M., Mckeown, N. and Shenker, S.: NOX: Towards an Operating System for Networks, *ACM SIGCOMM Computer Communication Review*, Vol.38, pp.105–110 (2008).

[12] Erickson, D.: Beacon Home, available from ⟨https://openflow.stanford.edu/display/Beacon/Home/⟩.

[13] Cai, Z., Cox, A.L. and Ng, T.S.E.: Maestro: A System for Scalable OpenFlow Control, available from ⟨http://www.cs.rice.edu/~eugeneng/papers/TR10-11.pdf⟩.

[14] Project Floodlight: Floodlight OpenFlow Controller, available from ⟨http://www.projectfloodlight.org/floodlight/⟩.

[15] Trema: Full-Stack OpenFlow Framework in Ruby and C, available from ⟨http://trema.github.com/⟩.

[16] Ryu SDN Framework Community: Ryu Web Site, available from ⟨https://osrg.github.io/ryu/⟩.

[17] PICA8 Inc: PICA8 Open Networking, available from ⟨http://www.pica8.com/⟩.

[18] Spirent Federal Systems: SmartBits 600B, available from ⟨http://www.spirentfederal.com/ip/products/smartbits/datasheets/⟩.

[19] Mo, W., He, J., Karbassian, M.M., Wissinger, J. and Peyghambarian, N.: Situation-aware Multipath Routing and Wavelength Re-assignment in a Unified Packet-circuit OpenFlow Network, *Proc. Optical Fiber Communication Conference and Exhibition/National Fiber Optic Engineers Conference* (*OFC/NFOEC'13*), pp.1–3 (2013).

[20] Egilmez, H.E., Civanlar, S. and Tekalp, A.M.: A Distributed QoS Routing Architecture for Scalable Video Streaming over Multi-domain OpenFlow Networks, *Proc. IEEE 19th International Conference on Image Processing* (*IEEE ICIP'12*), pp.1–4 (2012).

[21] 君山博之，北村匡彦，藤井竜也，丸山 充：OpenFlow スイッチを使った高精細映像動的仮想マルチパス伝送方式の提案，電子情報通信学会技術研究報告，コミュニケーションクオリティ研究会，Vol.115, No.496, pp.193–198 (2015).

[22] Staessens, D., Sharma, S., Colle, D., Pickavet, M. and Demeester, P.: Software Defined Networking: Meeting Carrier Grade Requirements, *Proc. IEEE 18th International Workshop on Local and Metropolitan Area Networks* (*IEEE LANMAN'11*), pp.1–6 (2011).

[23] Braga, R., Mota, E. and Passito, A.: Lightweight DDoS Flooding Attack Eetection using NOX/OpenFlow, *Proc. IEEE 35th Conference on Local Computer Networks* (*IEEE LCN'10*), pp.408–415 (2010).

[24] Das, S., Sharafat, A.R., Parulkar, G. and Mckeown, N.: MPLS with a Simple OPEN Control Plane, *Proc. Optical Fiber Communication Conference and Exhibition/National Fiber Optic Engineers Conference* (*OFC/NFOEC'11*), pp.1–3 (2011).

[25] 三宅信之，栗山俊通，田村孝之：OpenFlow と Open vSwitch による IP スプーフィング機能の実装，情報処理学会第 77 回全国大会講演論文集，pp.35–36 (2015).

[26] 原瑠理子，長谷川友香，小口正人：モニタリング情報に基づく OpenFlow を用いたネットワークトラフィック制御モデル，日本データベース学会第 6 回データ工学と情報マネジメントに関するフォーラム（DEIM'14），pp.1–3 (2014).

[27] Hu, S., Chen, K., Wu, H., Bai, W., Lan, C., Wang, H., Zhao, H. and Guo, C.: Explicit Path Control in Commodity Data Centers: Design and Applications, *Proc. 12th USENIX Symposium on Networked Systems De-*

sign and Implementation (*USENIX NSDI'15*), pp.15–28 (2015).
[28] Khurshid, A., Zou, X., Zhou, W., Caesar, M. and Godfrey, P.B.: VeriFlow: Verifying Networkwide Invariants in Real Time, *ACM SIGCOMM Computer Communication Review*, Vol.42, pp.467–472 (2014).
[29] Perlman, R., Eastlake, D., Dutt, D., Gai, S. and Ghanwani, A.: Routing Bridges (RBridges): Base Protocol Specification, RFC 6325 (2011).
[30] キットスワン，ナッタポン，大木英司：OpenFlow ネットワークにおけるトラフィック分散の実装，電子情報通信学会技術研究報告，ネットワークシステム研究会，Vol.116, No.111, pp.91–94 (2016).
[31] Handigol, N., Seetharaman, S., McKeown, N. and Johari, R.: Plug-n-Serve: Load-Balancing Web Traffic using OpenFlow, *ACM SIGCOMM Demo*, Vol.4, p.6 (2009).
[32] Yamasaki, Y., Miyamoto, Y., Yamato, J., Goto, H. and Sone, H.: Flexible Access Management System for Campus VLAN Based on OpenFlow, *Proc. IEEE/IPSJ 11th International Symposium Applications and the Internet* (*IEEE SAINT'11*), pp.347–351 (2011).
[33] 永山千博，長野純一，篠宮紀彦：通信リンクの使用状況を考慮したサイクルに基づく障害復旧方式の OpenFlow による実現，情報処理学会第 76 回全国大会講演論文集，pp.465–466 (2014).
[34] Bredel, M., Bozakov, Z., Barczyk, A. and Newman, H.: Flow-based Load Balancing in Multipathed Layer-2 Networks using OpenFlow and Multipath-TCP, *Proc. 3rd Workshop on Hot Topics in Software Defined Networking* (*ACM HotSDN'14*), pp.213–214 (2014).
[35] Egilmez, H.E., Gorkemli, B., Tekalp, A.M. and Civanlar, S.: Scalable Video Streaming over OpenFlow Networks: An Optimization Framework for QoS Routing, *Proc. 18th IEEE International Conference on Image Processing* (*IEEE ICIP'11*), pp.1–4 (2011).
[36] Laga, S., Cleemput, T.V., Raemdonck, F.V., Vanhoutte, F., Bouten, N., Claeys, M. and Turck, F.D.: Optimizing Scalable Video Delivery through OpenFlow Layer-based Routing, *Proc. IEEE Network Operations and Management Symposium* (*IEEE NOMS'14*), pp.1–4 (2014).
[37] Karl, M., Gruen, J. and Herfet, T.: Multimedia Optimized Routing in OpenFlow Networks, *Proc. 19th IEEE International Conference on Networks* (*ICON'13*), pp.1–6 (2013).
[38] Othman, O. and Okamura, K.: Design and Implementation of Application Based Routing Using OpenFlow, *Proc. 5th International Conference on Future Internet Technologies* (*CFI'10*), pp.1–8 (2010).
[39] Nakasan, C., Ichikawa, K., Iida, H. and Uthayopas, P.: A Simple Multipath OpenFlow Controller using topology-based algorithm for Multipath TCP, *Proc. PRAGMA Workshop on International Clouds for Data Science* (*PRAGMA-ICDS'15*), pp.1–8 (2015).
[40] Yap, K., Huang, T., Dodson, B., Lam, M.S. and Mckeown, N.: Towards Software-friendly Networks, *Proc. ACM 1st Asia-Pacific Workshop on Systems* (*ACM APSys'10*), pp.49–54 (2010).
[41] 吉村　悠，木村真乃介，佐藤健哉：サーバー負荷情報に基づく OpenFlow を用いた経路分散方式の提案，情報科学技術フォーラム講演論文集（FIT'14），Vol.14, No.4, pp.325–326 (2015).
[42] 前田達憲，阿部洋丈，加藤和彦：スループット予測による経路選択を用いた OpenFlow 環境での MPTCP 転送，情報処理学会研究報告，システムソフトウェアとオペレーティング・システム研究会，Vol.2013-OS-127, No.7, pp.1–5 (2013).
[43] 築地巧明，永田　晃，鶴　正人：OpenFlow ネットワーク上のマルチパス・マルチキャスト・ファイル転送，電子情報通信学会技術研究報告，コミュニケーションクオリティ研究会，Vol.112, No.476, pp.1–4 (2013).
[44] 木村　舟，シュウエンシン，カフレベド：SDN におけるアプリケーション毎最適経路選択手法，電子情報通信学会技術研究報告，情報ネットワーク研究会，Vol.115, No.370, pp.71–76 (2015).
[45] 篠原悠介，千葉靖伸，下西英之，本間裕大，合田雅樹：データセンタネットワークにおける効率的マルチパスルーティングとその OpenFlow ネットワークへの適用，電子情報通信学会技術研究報告，ネットワークシステム研究会，Vol.109, No.448, pp.13–18 (2010).
[46] 神山智行，淺谷耕一，水野　修：オープンフローネットワークにおけるマルチパスルーティング方式の適用性評価，電子情報通信学会技術研究報告，ネットワークシステム研究会，Vol.115, No.483, pp.417–422, 電子情報通信学会 (2015).
[47] Li, Y. and Pan, D.: OpenFlow based Load Balancing for Fat-Tree Networks with Multipath Support, *Proc. IEEE 12th International Conference on Communications* (*IEEE ICC'13*), pp.1–5 (2013).
[48] 西村俊彦，長坂康史，山木戸啓亮，堀　裕貴：OpenFlow を用いたデータ収集トラフィックの動的制御に関する研究，情報科学技術フォーラム講演論文集（FIT'15），Vol.14, No.4, pp.255–256 (2015).
[49] 鈴木洋司，高島正徳，岩田　淳：OpenFlow によるリンク帯域使用情報を用いたフローベースマルチパス帯域経路制御方式，電子情報通信学会総合大会講演論文集，p.41 (2010).
[50] Yu, M., Rexford, J., Freedman, M.J. and Wang, J.: Scalable flow-based networking with DIFANE, *ACM SIGCOMM Computer Communication Review*, Vol.40, pp.351–362 (2010).
[51] International Telecommunications Union: G.114: One-way transmission time, available from ⟨https://www.itu.int/rec/T-REC-G.114-200305-I/en⟩.
[52] Sidki, L., Ben-Shimol, Y. and Sadovski, A.: Fault tolerant mechanisms for SDN controllers, *2016 IEEE Conference on Network Function Virtualization and Software Defined Networks* (*NVF-SDN*), pp.173–178 (2016).
[53] Yelle, L.E.: The Learning Curve: Historical Review and Comprehensive Survey, *A Journal of the Decision Sciences Institute*, Vol.10, pp.302–328 (1979).

小野 悟　（正会員）

2010 年静岡大学大学院情報学研究科修士課程修了．2017 年静岡大学創造科学技術大学院自然科学教育部情報科学専攻博士後期課程修了．博士（情報学）．国立大学法人浜松医科大学に勤務．情報基盤センターにおいて，医療情報トータルシステムおよびキャンパスネットワークシステムの管理運営とネットワークシステムの研究に従事．

岩崎 裕輔 （学生会員）

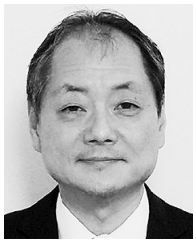
2017 年大阪大学工学部電子情報工学科卒業．現在，大阪大学大学院情報科学研究科修士課程に在学中．ワイヤレスセンシング，SDN，計算機ネットワークの研究に従事．

峰野 博史 （正会員）

1999 年静岡大学大学院理工学研究科修士課程修了．同年日本電信電話（株）入社．NTT サービスインテグレーション基盤研究所を経て，2002 年 10 月より静岡大学情報学部助手．2006 年九州大学大学院システム情報科学府博士（工学）．2011 年 4 月より静岡大学情報学部准教授．モバイル・センサネットワーク応用システムに関する研究に従事．電子情報通信学会，IEEE，ACM 各会員．

猿渡 俊介 （正会員）

2007 年東京大学大学院博士課程修了．科学博士．2003 年 IPA 未踏ソフトウェア創造事業，2006 年日本学術振興会学振特別研究員，2007 年イリノイ大学客員研究員，2008 年東京大学先端科学技術研究センター助教，2012 年静岡大学大学院情報学研究科助教（テニュアトラック）．2015 年同大学講師．2016 年より，大阪大学大学院情報科学研究科准教授．専門はワイヤレスネットワーク，センサネットワーク，システムソフトウェア等．2009 年電子情報通信学会論文賞．2010 年情報処理学会山下記念研究賞．2015 年より新エネルギー・産業技術総合開発機構技術委員．2016 年より内閣府政府調達苦情検討委員会専門委員．2017 年より総務省情報通信審議会専門委員．電子情報通信学会，IEEE，ACM 各会員．



渡辺 尚 （正会員）

1982 年大阪大学工学部通信工学科卒業．1984 年大阪大学大学院工学研究科通信工学専攻博士前期課程修了．1987 年大阪大学大学院工学研究科通信工学専攻博士後期課程修了．博士（工学）．同年徳島大学工学部助手．1990 年静岡大学工学部情報知識工学科助教授．2001 年静岡大学情報学部情報科学科教授．2013 年より，大阪大学大学院情報科学研究科教授．1995 年文部省在外研究員（カリフォルニア大学アーバイン校）．計算機ネットワーク，分散システムに関する研究に従事．2005 年より情報処理学会モバイルコンピューティングとユビキタス通信研究会主査．2011 年情報処理学会理事．2013 年電子情報通信学会知的環境とセンサネットワーク研究会副委員長．訳書『計算機設計技法』，『802.11 無線ネットワーク管理』等．電子情報通信学会，IEEE 各会員．本会フェロー．