2T-03

OpenFlow 医療情報ネットワークにおける動的経路選択方式の基礎検討

岩崎 裕輔 † 小林 真 ‡ 小野 悟 ‡† 猿渡 俊介 ‡ 渡辺 尚 ‡

† 大阪大学工学部 ‡ 大阪大学大学院情報科学研究科 ‡† 静岡大学創造科学技術大学院

1 はじめに

現在の医療情報ネットワークは様々なアプリケーションが生成するリッチなトラヒックを収容するために冗長かつ高コストな構成となっている場合が多い．筆者らは，医療情報ネットワークに OpenFlow[1] を導入することで低コストで高い性能と高信頼性を実現する OpenFlow 医療情報ネットワークの実現を目指している．本稿では OpenFlow 医療情報ネットワークにおける輻輳情報に基づく動的経路制御手法について述べる．

2 医療情報ネットワークの仮想化

電子カルテシステムの興隆と各種医療機器の高度化に伴って，医療情報ネットワークを流通するコンテンツは年々肥大化している．特に手術分野では，daVinci[2] などの医療ロボットシステムを利用した手術が増加しているため，大容量かつ低遅延なトラヒック要求が存在する．東日本大震災を契機に，災害時でも医療情報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．バックアップ対象ファイルは大容量であるため，ネットワークを高帯域に長時間専有してスケジュールされているタスクが予定通り終了しなかったり，他のジョブの遅延を招くなどの問題も抱えている [3]．

しかしながら，医療情報ネットワークにおいて高価で高性能な機器を導入しても性能が十分に活用されないという問題点がある．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るため，ピーク時の性能が常時要求されることは少ないからである．

本研究では，これらの課題を医療情報ネットワークの仮想化によって解決する．図 1 に現在の医療情報ネットワークの構成と仮想化した医療情報ネットワークの構成を示す．図 1 の左図のように，現在の医療情報ネットワークはツリー型のネットワーク構成である．多数のノードがエッジスイッチに，エッジスイッチはコアスイッチに接続されている．主要なノードを接続するリンクは冗長構成を取るためにマルチホームである場合が多く，コアスイッチに障害が発生すると全ての通信に障害が発生するため高機能なスイッチを用いる必要がある．

筆者らが実現する OpenFlow 医療情報ネットワークでは，図 1 の右図のように，コアスイッチとエッジスイッチを OpenFlow スイッチによるメッシュ型のネットワークに置き換える．従来のネットワークにおいて高スループットかつ高信頼なスイッチを使っていた部分をメッシュ型ネットワークにおける動的ルーティングで代替する．経路制御機能が一元化されることで，従来ノード毎に設定しなければならなかった経路設定が一括かつリアルタイムに実行可能となる．また，L1 から L4 間のパラメータを透過的に用いることでトラヒックに応じた動的な経路制御が可能となる．

3 輻輳情報を利用した経路選択手法

OpenFlow 医療情報ネットワークの実現に向けて，トラヒックに応じてフロー毎に動的に経路を割り当てる経路選択手法を実現した．具体的には，輻輳しているリンクや切断された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する．メッシュ型のトポロジにおいて動的に経路を選択するため，特定のリンクに障害が発生しても迂回するなどの高信頼性も実現できる．

3.1 ミクロな視点での動作

筆者らの OpenFlow 医療情報ネットワークでは，PathTable, FlowMap, TrafficTable の 3 つのデータを管理している．3 つのデータを適宜更新しながらフロー毎に経路を決定してパケットを配送する．

PathTable は OpenFlow 医療情報ネットワークで存在しうる全ての経路が記録されているテーブルである．スイッチの追加や離脱が発生する度に更新される．輻輳を回避する代替経路を取得する際に用いることを前提としているため，1 つの経路は同じリンクを 2 つ以上含まない．

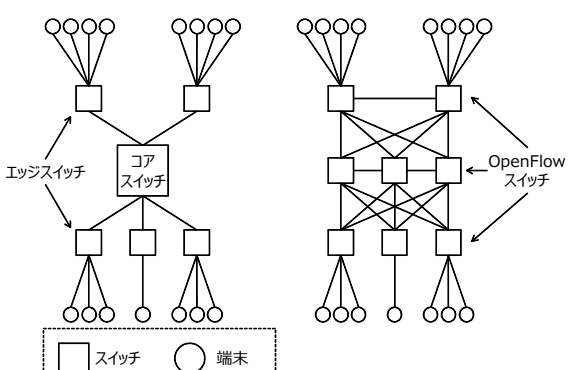


図 1: 現在の医療情報ネットワーク (左) と OpenFlow 医療情報ネットワーク (右)

Algorithm 1 congested イベントハンドラ

```
1: l ← congested link
2: P ← getPathSetFromFlowMap(l)
3: p ← choose a newest path from P
4: m ← get flow information from p
5: Q ← getPathSetFromPathTable(m)
6: Q = Q − p
7: while |Q| > 0 do
8:     q ← get a shortest path from Q
9:     if q does not include l then
10:        remove p from FlowMap
11:        add q to FlowMap
12:        break
13:    end if
14:    Q = Q − q
15: end while
```

FlowMap はフローと経路を関連付けるマップである．筆者らの OpenFlow 医療情報ネットワークでは，送信元 MAC アドレス，宛先 MAC アドレス，L3 プロトコルの種類，送信元 IP アドレス，送信元ポート，宛先 IP アドレス，宛先ポートの 7 つの情報の組を 1 つのフローとして扱っている．本稿ではこの 7 つの情報の組をフロー情報と呼ぶ．通信プロトコルによっては 7 つの情報を全て含んでいるとは限らないため，そのフローに存在しない項目は N/A として管理する．各スイッチでは OpenFlow の PACKET_IN イベントが発生すると，パケットからフロー情報を取得して，フロー情報をキーとして FlowMap から該当パケットを出力するポートを取得する．もし FlowMap に該当するレコードが存在しなかった場合，PathTable の中から該当フローを最短経路で配送可能な経路を取得して FlowMap に登録，取得した経路に該当する出力ポートからパケットを送信する．

TrafficTable は各リンクのトラヒック情報を記録・監視するテーブルである．OpenFlow の OFPMP_PORT_STATS を用いて，各リンクのトラヒックの情報を定期的に取得する．もしあるリンクで輻輳が発生していた場合には輻輳が発生したという congested イベントハンドラを実行する．

Algorithm 1 に congested イベントハンドラの動作を示す．congested イベントハンドラでは，輻輳が発生した場合に輻輳したリンクを使用しているフローを，輻輳していないリンクを含んだ経路に割り当てなおすことで各フローのスループットを向上する．getPathSetFromFlowMap(l) はリンク l を含む全ての経路を FlowMap から取得する関数，getPathSetFromPathTable(m) はフロー情報 m を配送可能な全ての経路を PathTable から取得する関数である．輻輳が発生した場合，まず，FlowMap の中から輻輳が発生したリンクを含む経路を全て取得する．その経路セットの中から最も最近割り当てられた経路を 1 つ選び，その経路のフロー情報を取得する．選んだ経路をここでは古い経路と呼ぶ．取得したフロー情報を用いてそのフローを配送可能かつ古い経路以外の経路を PathTable から全て取得する．取得した経路セットの中から経路が短い順に経路を 1 つ取得する．取得した経路が輻輳しているリンクを含まなかった場合，選んだ経路を新たな経路とする．取得した経路が輻輳しているリンクを含んでいた場合，経路セットから次に短い経路を 1 つ取得する．新たな経路が見つかった場合には PathMap から古い経路を削除，PathMap に新たな経路を追加する．

3.2 マクロな視点での動作

3.1 節の説明は実際の OpenFlow 上でのミクロな動作である．しかしながら，ミクロな動作だけでは 3.1 節の仕組みによって得られる効果を他の手法と比較しながら評価することは難しい．このような観点から，3.1 節の仕組みをネットワーク全体のマクロな仕組みで捉えたアルゴリズムを設計した．

Algorithm 2 に設計したアルゴリズムを示す．**Algorithm 2** は OpenFlow を用いた動作を前提としているため，1 つのフローに対して 1 つの経路を割り当てる．送信元端末と宛先端末の 2 つの情報の組をフローと定義する．getShortestPath(G, f) はグラフ G とフロー f よりグラフ G における送信元端末から宛先端末への最短路を取得する関数，getAllCombinations(L) はリンクセット L からリンクの全組み合わせの集合を要素数が多い順に返す関数である．フローセット F からフロー f を順番に取得する．フロー f が発生した場合，グラフ G における最短路 p_{shortest} を FlowMap に追加する．次に，輻輳しているリンクセット L を取得する．リンクセット L からリンクの全組み合わせの集合を取得

Copyright ©2017 Information Processing Society of Japan.
All Rights Reserved.

Algorithm 2 シングルパス割り当て手法

```
F ← flow set
G ← this network graph
for all f in F do
    p_shortest ← getShortestPath(G, f)
    add p_shortest to FlowMap
    L ← congested link set
    for all C ← getAllCombinations(L) do
        H ← get auxiliary graph by removing links C from G
        q ← getShortestPath(H, f)
        if q exists then
            remove p_shortest from FlowMap
            add q to FlowMap
            break
        end if
    end for
end for
```

後，それぞれの集合 C ごとに以下の操作を実行する．まず，輻輳が発生しているリンクの集合 C を取り除いた補助グラフ H を作成する．補助グラフ H から最短路 q を取得する．q が存在するならば，p_{shortest} を FlowMap から削除を行い，q を新しく FlowMap に追加する．

4 評価

4.1 評価環境

図 2 に本研究に用いるネットワークのメッシュ型のトポロジを示す．各ネットワークは OpenFlow スイッチと端末で構成されている．数字のノードが OpenFlow スイッチ，英字のノードが端末を表している．ネットワークにおける各リンクの容量は 1 [Gbps] としている．

提案手法の性能を相対的に評価するために，比較対象として multipath，brute force，shortest path を用いた．multipath は提案手法を 1 つのフローに対して 2 つ以上の経路を割り当てられるように拡張した手法である．仮に 1 つのフローを複数の経路に割り当てることができたと仮定した場合の性能を評価することを目的としている．OpenFlow ではフロー単位でパケットをルーティングするため，1 つのフローに対して 1 つの経路しか割り当てることができないというハードウェア的な制約がある．

brute force は，マルチパスを含む全ての経路組み合わせを総当たりして総スループットが最大となる経路を割り当てる手法である．膨大な計算量を必要とするので実用的では無い．達成可能なスループットの最大値を示す指標となる．

shortest path は，各フローに最短路を割り当てる手法である．輻輳が発生しても輻輳を回避しないためスループットが低くなることが予想される．提案手法の輻輳回避の性能を示す指標となる．

4.2 固定フローに対する評価

提案手法の定性的な違いを評価するために，経路の割り当てが異なると想定されるフローを意図的に生成させて評価を行った．フローセットとして，フローセット 1 (a → c, b → d)，フローセット 2 (a → c, a → e, b → d)，フローセット 3 (a → c, b → d, a → e) を用いた．

フローセット 1 を shortest path で割り当てた場合，a → 1 → 2 → c (0.50 [Gbps])，b → 1 → 2 → d (0.50 [Gbps]) の 2 本の経路が割り当てられる．フローセット 1 を提案手法で割り当てた場合，a → 1 → 2 → c (1.00 [Gbps])，b → 1 → 3 → 2 → d (1.00 [Gbps]) の 2 本の経路が割り当てられる．フローセット 1 において，shortest path で割り当てた場合の総スループットは 1.00 [Gbps]，提案手法で割り当てた場合の総スループットは 2.00 [Gbps] となる．提案手法が shortest path よりも 2 倍の高い性能を実現している．

フローセット 2 を提案手法で割り当てた場合，a → 1 → 2 → c (0.50 [Gbps])，a → 1 → 3 → e (0.50 [Gbps])，b → 1 → 2 → d (0.50 [Gbps]) の 3 本の経路が割り当てられる．フローセット 2 を multipath で割り当てた場合，a → 1 → 2 → c (0.50 [Gbps])，a → 1 → 3 → e (0.50 [Gbps])，b → 1 → 2 → d (0.50 [Gbps])，b → 1 → 3 → 2 → d (0.50 [Gbps]) の 4 本の経路が割り当てられる．フローセット 2 において，提案手法で割り当てた場合の総スループットは 1.50 [Gbps]，multipath で割り当てた場合の総スループットは 2.00 [Gbps] となる．multipath が提案手法よりも約 1.3 倍の高い性能を実現している．

フローセット 3 を multipath で割り当てた場合，a → 1 → 2 → c (0.50 [Gbps])，b → 1 → 3 → 2 → d (0.50 [Gbps])，a → 1 → 3 → e (0.50 [Gbps]) の 3 本の経路が割り当てられる．フローセット 3 を brute force で割り当てた場合，a → 1 → 2 → c (0.50 [Gbps])，b → 1 → 4 → 2 → d (1.00 [Gbps])，a → 1 → 3 → e (0.50 [Gbps]) の 3 本の経路が割り当てられる．フローセット 3 において，multipath で割り当てた場合の総スループットは 1.50 [Gbps]，brute force で割り当てた場合の総スループットは 2.00 [Gbps] となる．brute force が multipath よりも約 1.3 倍の高い性能を実現している．

4.3 ランダムなフローに対する評価

4.2 節では，各手法に差が出るフローを意図的に生成して評価を行った．各手法を相対的に比較することを目的として，ランダムに生成したフロー下においてフロー数が変化した場合の総スループットを評価した．

図 3，図 4 に，メッシュトポロジにおけるフロー数に対する改善率を示す．改善率は shortest path に対する総スループットの上昇率である．T をそれぞれのアルゴリズムにおける総スループット，T_{shortest} は shortest path における総スループットとすると，改善率は $\frac{T}{T_{\text{shortest}}} - 1$ となる．図 3 はフロー数を 1 から 5 まで変化させた評価，図 4 はフロー数を 1 から 100 まで変化させた評価である．計算時間の問題から，brute force はフロー数 5 までの評価を行った．

図 3，図 4 より，以下の 3 つのことが分かる．1 つ目は，提案手法は shortest path よりも性能が高いことである．提案手法の有効性が確認できたと考えている．2 つ目は，brute force が最も高い性能を示していることである．multipath では最適な割り当てにならない場合があることが分かる．3 つ目は，フロー数が増加するにしたがって各手法の差が小さくなり，フロー数が 100 の時は性能に差がなくなることが分かる．本評価ではフローをランダムに生成しているため，フロー数が増えれば最短経路で経路を選択していたとしても経路が分散されているからだと考えられる．医療情報ネットワークでの現実のトラヒックを用いての評価が求められる．

4.4 計算量の評価

スループットの評価では brute force が最も高い性能を発揮していた．しかしながら，実際の OpenFlow 医療情報ネットワークでは時々刻々と変化するトラヒックに対して動的に経路を制御するため，計算量は少ないことが望ましい．このような観点から，送信元端末と宛先端末がランダムに選択されるフローにおける計算時間の評価を行った．図 5 にメッシュトポロジにおけるフロー数に対する計算時間を示す．提案手法が shortest path と同程度の計算時間で動作する事が分かる．提案手法が経路探索を最短路と輻輳情報のみを用いて経路を割り当てることで，少ない計算量で経路を算出できているからである．一方で，フロー数の増加に伴い brute force に必要とする計算量は指数関数的に増加していることが分かる．

5 おわりに

本稿では，医療情報ネットワークにおける OpenFlow を用いた動的経路選択手法を提案した．現在，実機実装および実際の医療情報ネットワークで発生するフロー情報での評価を行っている．

謝辞 本研究は JSPS 科研費 JP16H01718 の助成を受けたものである．

参考文献 [1] N. McKeown, et al., OpenFlow: enabling innovation in campus networks. In *ACM SIGCOMM Computer Communication Review*, Vol. 38, pp. 69–74, 2008. [2] N. G. Hockstein, et al., Robotic microlaryngeal surgery: A technical feasibility study using the davinci surgical robot and an airway mannequin. *Laryngoscope*, Vol. 115, pp. 780–784, 2005. [3] 小野ほか．大容量通信に対し動的にスケールする医療情報ネットワーク仮想化技術の検討．信学ソ大，2016.

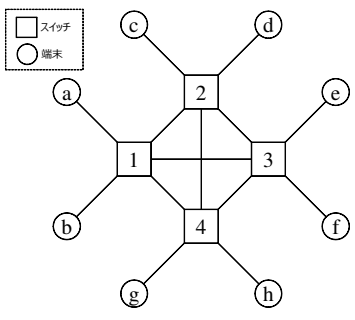


図 2: メッシュトポロジ

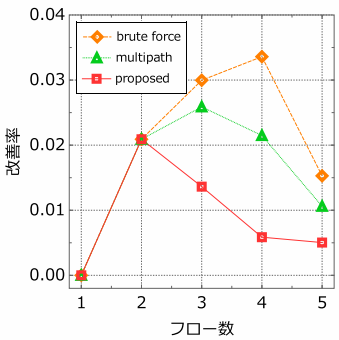


図 3: フロー数に対する改善率

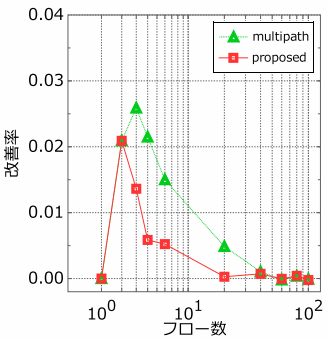


図 4: フロー数に対する改善率

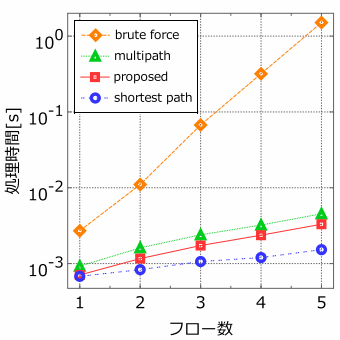


図 5: フロー数に対する計算時間

Copyright ©2017 Information Processing Society of Japan.
All Rights Reserved.