情報処理学会論文誌 Vol.60 No.2 591–607 (Feb. 2019)

省電力化のための MPTCPによるSDNを用いたスイッチ間帯域切替え方式

西口 雅人[1,†1,a)] 木村 成伴[2,b)]

受付日 2018年3月30日, 採録日 2018年11月7日

概要：近年，通信機器の高速化がすすむ一方で，その消費電力を削減することは重要な課題となっている．しかし，通信中のリンクを対象とした従来の省電力化手法には課題があった．そこで本論文では，SDN（Software Defined Network）スイッチで構築された広域ネットワークにおいて，このうちの1台のスイッチにつながるクライアントが，インターネットを含む外部ネットワークとの境界に置かれた1台のスイッチを経由して，サーバと MPTCP によるマルチパス通信を行う環境を対象とする．そして，クライアントとサーバ間に，低帯域，高帯域に加えて，通信中断なく切替え可能な新たな中間帯域を提供する，スイッチ間帯域切替え方式を提案する．最後に，提案方式を実装したネットワークにおいて通信実験を行い，その通信中断時間やスループット，消費電力を評価し，特に，各スイッチの積算消費電力は，1日あたり合計で最大約 60 Wh（電力比では 1.35%）削減できることを示す．

キーワード：省電力化，adaptive link rate，MPTCP，SDN，OpenFlow

Adaptive Link Rate Switching Using MPTCP in SDN-based Networks for Power Savings

Masato Nishiguchi[1,†1,a)] Shigetomo Kimura[2,b)]

Received: March 30, 2018, Accepted: November 7, 2018

Abstract: In recent years, the speed of network devices has rapidly increased, but it is also a critical issue to reduce the power consumption of these devices. However, conventional methods for power savings of communication links during transmission have problems. To solve the problems, this paper supposes wide area networks constructed by SDN (Software Defined Network) switches where clients connect to one of the switches and communicate over multipath connections by MPTCP with a server connected at an external network including the Internet through another switch that is placed on the boundary between the switch and the external network. Then, this paper proposes a new adaptive link rate switching method to provide in addition to low and high rates, a middle rate that MPTCP connections can switch to uninterruptedly. Finally, the communication experiments based on the implementation of the proposed method evaluate the communication interruption time, throughput, and power consumption, and show the total power consumption per a day of each switch can reduce up to about 60 Wh (1.35% of the power ratio) in especial.

Keywords: power savings, adaptive link rate, MPTCP, SDN, OpenFlow

1. はじめに

近年，インターネットトラフィックが急増し，ネットワーク通信におけるリンク帯域拡大の需要が高まっている．総務省の調査によれば，2017 年 11 月時点での国内の総ダウ

[1] 筑波大学大学院システム情報工学研究科
Graduate School of Systems and Information Engineering, University of Tsukuba, Tsukuba, Ibaraki 305–8573, Japan
[2] 筑波大学システム情報系情報工学域
Faculty of Engineering, Information and Systems, University of Tsukuba, Tsukuba, Ibaraki 305–8573, Japan
[†1] 現在，日本電信電話株式会社 NTT ネットワークサービスシステム研究所
Presently with NTT Network Service Systems Laboratories, NTT Corporation
[a)] masato.nishiguchi.ty@hco.ntt.co.jp
[b)] kimura@netlab.cs.tsukuba.ac.jp

ンロードトラフィックは推定で約 10.8 Tbps であり，これは前年同月比で 31.6%の増加となっている [1]．今後，この傾向はより顕著となり，リンク帯域拡大の需要はますます高まると考えられる．

その一方で，通信機器の省電力化は重要な課題であり，特に消費電力の全体に占める割合の高い，高速化された通信機器の消費電力を削減することは，省電力化の観点から効果的であることから，様々な省電力化手法が提案されている．その代表的なものの 1 つに，リンクが通信を行っていないときに，リンクを消費電力の少ない状態に遷移させることで省電力化を図る LPI（Low Power Idle）がある．また，他の手法として，TE（Traffic Engineering）によって通信に使用するパスを集約し，非通信のリンクをリンクオフにすることで省電力化を図る方式 [2], [3] などもある．このような非通信のリンクを対象とした手法は，実装が容易であり，通信に与える影響も小さいため，たとえば，LPI は，従来の Ethernet に省電力性能を持たせた，IEEE 802.3az 標準の EEE（Energy Efficient Ethernet）で採用されている [4]．しかし，インターネットトラフィックが大規模化・複雑化する昨今のネットワーク通信においては，特定のリンクを通信で使われないようにすることが難しくなることから，これらの手法の省電力性能が低下する可能性がある．実際，LPI では，状態遷移が多くなると省電力性能が著しく低下するという問題が報告されている [5]．

また，通信中のリンクを対象とした省電力化手法として，リンクの帯域を通信状況に応じて切り替えることで省電力化を図る ALR（Adaptive Link Rate）がある．この方式では，Ethernet を対象としており，帯域にともなって消費電力が増加する特性 [6] から，リンクの帯域を低帯域と高帯域の 2 つに設定し，それらを通信状況に応じて切り替えることで省電力化を図るが，切替えの際に発生する通信中断が大きいという問題があった [7]．この問題を改善するための特殊なハードウェア機構も提案・実装されている [8] が，既存の通信機器との入れ替えをともなうため，普及面に課題があり，実用化には至っていない．

これに対して著者らは，端末が，無線 LAN と Gigabit Ethernet など，消費電力の異なる複数のネットワークインタフェースを持つ環境を仮定し，通信時に，宛先 IP アドレスに応じて使用するリンクを動的に切り替える手法を提案し，切替えにともなう通信中断を発生させることなく，省電力化が達成できることを示した [9]．しかし，省電力化の対象が端末に限られることから，その成果の適用範囲をネットワーク全体に拡大することが求められていた．

そこで本論文では，SDN（Software Defined Network）スイッチで構築された広域ネットワークにおいて，ALR を適用することを目標とする．SDN は，セキュリティやネットワーク管理の柔軟化を目的として，キャンパスネットワークや社内ネットワーク，データセンタなどにおいて導入がすすみ，最近では，WAN への適用もなされている．こうした背景から，本論文では，SDN の今後のさらなる導入拡大を見据え，SDN が導入されたネットワークにおいて機器の変更を必要としない省電力化を図る．この目的のため，本論文では，キャンパスネットワークや社内ネットワーク，ISP 間ネットワークといった，十台～数十台程度のスイッチが接続された中～大規模の平均リンク使用率が低いネットワークで，すべてのルータが SDN に対応したスイッチで構成されており，それらが単一のコントローラによって制御されるものを対象とする．そして，このうちの 1 台のスイッチにつながるクライアントが，インターネットを含む外部ネットワークとの境界に置かれた 1 台のスイッチを経由して，サーバと MPTCP によるマルチパス通信を行う状況を仮定する．この条件下において，クライアントとサーバ間に，ALR における低帯域と高帯域に加えて，通信中断なく切替え可能な新たな中間帯域を提供する，スイッチ間帯域切替え方式を提案する．ただし，この中間帯域は，ALR のように単にリンク帯域を切り替えるのではなく，高帯域リンクの消費電力を超えないだけの本数分の低帯域リンクを，MPTCP と SDN によるマルチパス通信で束ねて帯域を拡大することで実現する．この方式により，通信状況に応じた過剰ではない帯域と消費電力を実現することで，通信中断をともなう帯域切替え回数の削減と省電力性能の向上を可能にする．

本論文の構成は以下のとおりである．まず，2 章で，既存のリンク省電力化手法とその課題について述べる．3 章では，提案方式で用いる MPTCP と SDN，および SDN を用いた省電力化手法を関連研究として説明する．4 章では，本論文で提案するスイッチ間帯域切替え方式について述べる．そして，3 台の SDN スイッチで構築したネットワークに本方式を実装し，この上で通信実験を行うことで，既存の ALR を用いた場合と通信中断時間やスループット，消費電力を比較し，本方式の有効性を確認する．特に，各スイッチの積算消費電力は，1 日あたり合計で最大約 60 Wh（電力比では 1.35%）削減できることを示す．また，通信実験の結果から考察・改善案の検討およびその評価を行う．最後に，5 章で本論文をまとめ，今後の課題を述べる．

2. リンクの省電力化

本章では，既存のリンク省電力化手法について示す．まず，前提知識となる Ethernet について述べた後，非通信のリンクおよび通信中のリンクを対象にしたリンク省電力化手法についてそれぞれ説明する．

2.1 Ethernet

Ethernet の伝送規格のうち，現時点で広く普及している，ツイストペアケーブルを用いたものとして 100BASE-TX，1000BASE-T，10GBASE-T などがある．Ethernet の NIC

（Network Interface Card）では，これらのうちから複数の伝送規格をサポートしているのが一般的で，使用する伝送規格は，ソフトウェアによって任意に切り替えることが可能である．また，NIC やこれを接続するスイッチングハブには，接続時に双方で利用可能な最大の帯域を持つ伝送規格が自動的に選択されるオートネゴシエーション機能を持つものがあり，1000BASE-T 以降では，この機能を持つことが必須とされている．

一般に，帯域の増加にともなって消費電力も増加することから，たとえば，100BASE-TX は 1000BASE-T と比べて 3 W，10GBASE-T と比べて 18 W の電力差があることが報告されている [6]．さらに，文献 [4] では，1000BASE-T と 10GBASE-T には 10 倍の電力差があるとしている．

2.2 非通信のリンクを対象にした手法

非通信（通信を行っていない状態）のリンクを対象にした省電力化手法は，リンクレベルでの省電力化とネットワークレベルでの省電力化に分類される．

2.2.1 リンクレベルでの省電力化

リンクレベルでの省電力化として，非通信のリンクをスリープ状態にさせることで省電力化を図る手法 [10], [11] が提案されていたが，スリープ状態からアクティブ状態への遷移に時間がかかることや，状態遷移が多くなると機器の劣化を促進させるといった問題があった．

これを改善するため，LPI では，アイドル状態を新たに追加した．この状態では，パケットの送受信は行われておらず，消費電力はアクティブ状態の 10%程度に削減される [4] が，通信先とのリンクを維持するためのリフレッシュ動作を一定の時間間隔で行う．これによって，スリープ状態よりも消費電力が高くなるものの，アクティブ状態へ遷移する時間が短くなり，通信に与える影響を抑えつつ，省電力化を図ることができる．

しかし，その一方で，状態遷移が多くなると，LPI の省電力性能が著しく低下するという問題が報告されている [5]．この報告によると，パケットの到着がポアソン過程に従う場合，100BASE-TX では 20%の負荷（パケットの平均到着率が通信容量の 20%）で最大消費電力の 50%程度の電力消費であるのに対し，1000BASE-T では 100%に近い電力を消費するなど，LPI による省電力化は，状態遷移を頻繁に引き起こすトラフィックにより，その効果が小さくなることが分かる．これを改善するため，Coalescing LPI [4], [12] では，キューにパケットをある程度バッファしてからバースト転送する機能を LPI に追加し，まとまった非通信時間を確保することで，LPI の省電力性能の安定化を実現した．しかし，パケットのバッファリングにともなう遅延を招くため，省電力性能とスループットがトレードオフとなっている．

2.2.2 ネットワークレベルでの省電力化

前項で述べたリンクレベルでの省電力化を，TE によってネットワーク全体に拡張したものに，文献 [2], [3], [13] などがある．

文献 [2], [3] では，トラフィックを必要最小限のリンクに集約し，それ以外の非通信となったリンクをリンクオフさせることで，ネットワーク全体の省電力化を図る，MiDORi（Multi-(layer, path, and resources) Dynamically Optimized Routing）ネットワークとそのためのパス計算方式を提案している．しかし，プロトタイプによる評価ではトポロジの再構成に大きな時間がかかっており，また，特殊なハードウェア機構やプロトコルを要するため，既存のネットワークへの適用は難しいと考えられる．

文献 [13] も同様に，中央集権サーバが各ルータからトラフィック情報を収集し，ヒューリスティックな計算によりトポロジを求めるが，その情報収集は既存の OSPF（Open Shortest Path First）プロトコルのリンク状態広告を用いることで，既存のネットワークへの適用を容易にしている．しかし，中央集権サーバが各ルータの動作状態を直接制御するわけではないため，中央集権サーバの指令によって非通信（トラフィックが流れない状態）とすることができても，それによって，リンクがスリープ状態やアイドル状態などの低電力状態に遷移するかどうかはルータに搭載されたネットワークインタフェースに依存するため，省電力性能がサーバの想定どおりに達成できるとは限らない．また，OSPF のリンク状態広告は flooding により配信するため，トラフィックの変動が激しいネットワークでは，その変動に追従するように頻繁に flooding した場合，ネットワークの混雑を引き起こすと考えられる．さらに，リンクの状態変更の頻度が多くなれば，経路再計算が与える各ルータのプロセッサへの負荷の増大も避けられない．

2.3 通信中のリンクを対象にした手法

通信中のリンクを対象にした手法として，以下では ALR とこれを改良した著者らの従来研究について述べる．

2.3.1 ALR（Adaptive Link Rate）

ALR は，主に PC とスイッチ間のリンクを対象とし，ネットワークのトラフィック量などに応じて，リンクの帯域を動的に切り替えることで省電力化を図る．

まず，文献 [7], [8], [14], [15], [16] は，1000BASE-T を対象とした ALR であり，通常は 100BASE-TX の状態で消費電力を削減するが，高負荷時には 1000BASE-T に切り替えて高速に通信する．

文献 [14] では，ALR のための MAC ハンドシェークプロトコルを提案している．文献 [15] では，NIC のバッファに届いたパケット量を切替え条件とし，さらに，文献 [16] では，リンクの使用率を監視し，インタフェースから一定時間に転送されたバイト数も切替えの条件としている．文

献 [7] では，帯域切替え後，一定時間は切替え後の帯域を保つことで，帯域が頻繁に変更されないようにしている．また，文献 [8] では，実際に ALR を適用したハードウェアのプロトタイプを作成し，通信実験によって帯域切替え時間や消費電力を測定している．

一方，100G Ethernet は，現時点では光ファイバを用いた方式しか標準化されておらず，ツイストペアケーブルを対象とした 1000BASE-T とは構造が違うため，ALR の実現方法も異なっている．たとえば，文献 [17] では，100G Ethernet が，マルチレーンと呼ばれる，複数の伝送路で並列に伝送していることに着目し，この伝送路の数を状況に応じて減らすことで省電力化を実現している．

以上の方式では，帯域切替えの判断を，バッファに届いたパケット量やリンク使用率のみで行っている．このため，切替え時にネットワーク層以上の情報が得られず，ノードがどのような通信を行っているのかという情報を考慮した帯域切替えができない．

また，異なる伝送規格の帯域の差は，たとえば，100BASE-TX と 1000BASE-T の間のように，最大で 10 倍の開きがある．このため，リンク使用率が徐々に高まり，ALR によって低帯域から高帯域に切り替えたとしても，その後，ある程度の期間は未使用の帯域が多く存在すると考えられる．実際，コアネットワークでの平均リンク使用率は 30〜40%程度 [18]，ネットワークエッジでは 5%以下であると報告されている [7], [8]．さらに，Ethernet の消費電力は，リンクの使用率にはほとんど影響されず，その帯域によって決定される [7] ため，低帯域から高帯域への切替え後に，未使用の帯域が多く存在するのであれば，その消費電力は使用する帯域に見合った量であるとはいえない．この問題は，低帯域と高帯域の間の帯域と，帯域の大きさに見合った段階的な消費電力を提供することで解決できると考えられ，提案方式では，複数の低帯域リンクによるマルチパス通信を用いてこれを実現する．

2.3.2 宛先 IP アドレスを考慮したリンク切替え方式

著者らは，ALR がかかえる問題点を解決するため，端末が，無線 LAN と Gigabit Ethernet など，消費電力の異なる複数の NIC を持つ環境を仮定し，通信時に，宛先 IP アドレスに応じて使用するリンクを動的に切り替える，リンク切替え方式を提案し，その有効性を示した [9]．

リンク切替え方式では，小規模なオフィスや家庭内での典型的なネットワークを適用対象としており，このようなネットワークにおける，通信先のサブネットによる通信速度と使用リンクによる消費電力の差異に着目し，クライアントが，送信先のサブネットによって使用するリンクを切り替えることで消費電力を抑える．すなわち，通常は，無線 LAN などの低速だが消費電力の低いリンクで接続し，同じ LAN 内の相手など，エンド–エンドで高速に通信できる通信先と通信する場合は，Gigabit Ethernet などの消費電力は高いが高速なリンクで，短時間に通信を完了させることで，消費電力は低いが低速なリンクで長時間通信するよりも，消費電力を抑えることが可能となる．

さらに，リンクの切替えには，ルーティングテーブルのメトリック値と Gigabit Ethernet のオートネゴシエーション機能を用いることで，既存のハードウェアを変更せずに切替えにともなう通信中断を防ぐことが可能である．具体的には，あらかじめ，クライアントのルーティングテーブルにおいて，Gigabit Ethernet のメトリック値を無線 LAN よりも小さい値に設定しておき，消費電力を抑えるため，通常は，Gigabit Ethernet をリンクダウンさせておく．その後，LAN 内の相手などと通信を行うときは，クライアントは Gigabit Ethernet をリンクアップさせるが，接続先のスイッチとオートネゴシエーションを行っている間は，メトリック値の大きな無線 LAN での通信が継続される．オートネゴシエーションが完了した後は，メトリック値の小さな Gigabit Ethernet に，通信中断なく自動的に切り替わる．そして，実機による通信実験の結果，常時単一のリンクで接続した場合に比べて，クライアントや通信機器（ルータ，スイッチ，アクセスポイント）の消費電力を削減できることが確認できた．

しかし，リンク切替え方式は，小規模ネットワークにおける通信機器の省電力化を可能としたが，その適用範囲をネットワーク全体に拡大することが求められていた．そこで本論文では，SDN スイッチで構築された広域ネットワークにおいて，ALR を適用することを目標とし，省電力化のための MPTCP による SDN を用いたスイッチ間帯域切替え方式を提案する．次章では，その前提となる MPTCP と SDN について述べる．

3. 関連研究

本章では，提案方式で用いる MPTCP と SDN について述べた後，提案方式の関連研究として，SDN を用いた省電力化手法について説明する．

3.1 MPTCP

MPTCP は，TCP を拡張する形で策定されたトランスポート層のプロトコル [19], [20] であり，MPTCP コネクションは，複数のサブフローと呼ばれる TCP コネクションが束になって形成される．そして，これらのサブフローが複数の異なるパスを通ることによりマルチパス通信が実現する．また，MPTCP では，サブフローを TCP で実現しているため，見かけ上は TCP による通信と変わらないことが大きな特徴となる．このため，ミドルボックスのファイアウォールや NAT を変更しなくても，パケットが破棄されるのを回避でき，さらに，MPTCP に対応しない既存のアプリケーションと通信する場合でも，単一のサブフローのみを用いることで通信を継続することが可能と

なる．

Linux カーネル実装 [21] では，3 つのサブフローの割当て方がサポートされている．default はアドレス，つまり 1 つの NIC につき 1 つのサブフローを割り当てる方法である．次に，ndiffports は，1 つのポートにつき 1 つのサブフローを割り当てる方法であり，たとえシングルホーム環境下で，単一のアドレスしか持たない場合であっても，マルチパス通信を可能にする．そして，fullmesh では，default と ndiffports をあわせ，アドレスとポートの組につき 1 つのサブフローを割り当てている．しかし，たとえば，default による割当てにおいて，全 NIC が同じサブネットに接続されている場合や ndiffports による割当ての場合，すべてのサブフローが同じ経路をたどることになり，本来のマルチパス通信にはならない．これは，ネットワーク層が行う経路制御の働きにより，MPTCP のサブフローの違いにかかわらず，宛先 IP アドレスのみによって経路が決定されるため，同じサブネットから送られた同一 IP アドレス宛のパケットは，同じ経路をたどることになるためである．

提案方式では，主に ndiffports によってサーバと接続することを想定しており，これらのサブフローの経路を制御するため，次節で述べる SDN を用いる．

3.2 SDN（Software Defined Network）

SDN は，通信機器の制御機能と転送機能を分離し，制御機能をコントローラと呼ばれる機器が中央集権的に担うネットワークのことであり，この SDN を実現するために用いられる標準的な技術が OpenFlow である．

OpenFlow では，ネットワーク制御を行う OpenFlow コントローラと，データ転送を行う OpenFlow スイッチからネットワークが構成され，コントローラとスイッチ間の通信を規定する OpenFlow プロトコルが定義されている．コントローラは，OpenFlow プロトコルを用いてスイッチのフローテーブルにフローエントリ（パケットの転送ルール）をインストールし，スイッチはテーブルを参照してデータ転送を行う．フローエントリを構成する要素は，条件に合致するパケットに対して指定した操作を行う，マッチアンドアクション形式であり，その内容は OpenFlow のバージョンによって異なる．

本論文で用いる OpenFlow 1.0 のフローエントリは，ヘッダフィールド，カウンタ，アクションの 3 つの要素から構成される．ヘッダフィールドにはマッチの条件が記述され，その条件に合致したパケットに対してアクション部分で指定した操作が行われる．また，カウンタには条件にマッチしたパケットの統計情報が記録される．フローエントリには，データリンク層からトランスポート層までの情報を自由に組み合わせた条件が指定できるため，たとえば，文献 [22] では VLAN を，文献 [23] ではポート番号と IP アドレスを用いて，サブフローごとに異なる経路を与えることで，MPTCP によるマルチパス通信を実現している．

このフローエントリは，新規スイッチがネットワークに接続されたときなどにコントローラがあらかじめインストールする．しかし，対応するフローエントリの存在しないパケットをスイッチが受け取った場合は，スイッチ自身を示す固有の識別子（Datapath ID）やパケットを受信したポートの番号などの情報をパケットに加えた，Packet In メッセージをスイッチがコントローラに送り，これを受け取ったコントローラが，パケットの転送方法を示したフローエントリをそのスイッチにインストールを行うことで対応する．また，このとき，コントローラはスイッチに指定したポートからパケットを出力させることができる．この処理は Packet Out と呼ばれ，Packet In にともなって実行するほかにも，コントローラ自身でパケットを作成し，それをネットワークに流す場合などにも使用できる．

3.3 SDN を用いた省電力化手法

SDN を用いた既存の省電力化手法として，文献 [24], [25] では，2.2.2 項で述べた中央集権サーバを用いたトラフィック制御を，OpenFlow コントローラが担う形で省電力化を実現している．しかし，スイッチのスリープ状態とアクティブ状態の切替えのみを行うため，動作状態を頻繁に切り替える場合，その省電力性能が悪化する可能性がある．

これに対して，文献 [26] では，SDN を用いて ALR を実現する方式を提案している．この方式では，2 章で述べた既存の省電力化手法を以下の 3 つのレベルに分類し，それぞれのレベルごとの省電力化手法を，OpenFlow ネットワークの環境で実現している．

(1) チップレベル — ALR による省電力化
(2) ノードレベル — LPI による省電力化
(3) ネットワークレベル — TE による省電力化

このうち，チップレベルでの省電力化では，OpenFlow スイッチ間に帯域の異なる 2 本のリンクを接続し，それらをトラフィック量に応じて切り替える手法を採用している．そして，シミュレーションによる通信実験の結果，本方式を用いない状態と比べて，消費電力を削減できることを示している．しかし，Ethernet はスリープ状態であっても電力を消費するが，本シミュレーションではその点について考慮されていない．このため，実機で同様の実験をした場合，同様の削減結果が得られないことが懸念される．また，ALR によって低帯域から高帯域に切り替えても，2 つの帯域には最大で 10 倍の開きがあるため，切替え後，ある程度の期間は未使用の帯域が多く存在し，消費電力が余分にかかるという問題は解決されていない．

本章で述べたいずれの方式も，実機に適用する際に必要となる前提条件を十分に考慮していないという問題もある．たとえば，OpenFlow ではポートの動作状態を制御する機能をサポートしていないため，ポートのスリープやリ

ンクの帯域切替えは，SNMP や SSH など，OpenFlow プロトコル以外のプロトコルを通じて行う必要があり，これらの制御メッセージによる影響も考慮する必要がある．

さらに，シミュレーション環境では消費電力の正確な計測が難しいことから，実際の OpenFlow ネットワークを用いた消費電力の計測実験を行うことが求められる．そこで，次章では，ALR における低帯域と高帯域に加えて，通信中断なく切替え可能な新たな中間帯域を提供する，スイッチ間帯域切替え方式を提案する．そして，実機で構築したネットワークに本方式を実装し，これを用いた通信実験により，提案方式の有効性を確認する．

4. 提案方式

本章では，提案方式であるスイッチ間帯域切替え方式について説明する．そして，実機による通信実験を行い，提案方式の改善案の検討およびその評価を行う．

4.1 概要

提案方式では，**図 1** のようなネットワークを想定する．ここで，キャンパスネットは省電力化の対象となるネットワークであり，この中の 1 台のスイッチを通して，インターネットなどの外部ネットワークに接続されている．別の 1 台のスイッチには，学部などの拠点で利用するネットワークが接続されている．そして，一般ユーザが使用するクライアントは，このネットワークに接続され，キャンパスネットを通じて，外部ネットワークにあるサーバと MPTCP によるマルチパス通信を行う．すなわち，キャンパスネットには，始点と終点となるエッジスイッチが 2 つあり，サブフローが，いずれか一方のスイッチから入り，他方から出て，通信をする状況を仮定する．ただし，簡単のため，本論文では，外部ネットワークに接続するスイッチは 1 台に限るものとし，各スイッチに装備されている NIC はすべて同一のものであるとする．また，複数の拠点ネットワークがそれぞれ異なるスイッチにつながっていてもかまわないが，拠点ネットワーク間での通信はしないものとし，通信相手のサーバは外部ネットワークにあると限定する．

この条件下において，提案方式では，クライアントとサーバ間に，ALR における低帯域と高帯域に加えて，通信中断なく切替え可能な新たな中間帯域を提供する．この中間帯域は，ALR のように単にリンク帯域を切り替えるのではなく，総消費電力が，高帯域リンクの消費電力を超えない本数までの低帯域リンクを，MPTCP と SDN によるマルチパス通信で束ねて帯域を拡大することで実現する．そして，通信状況に応じてこれらの帯域を切り替え，ALR では過剰に提供されていた帯域と消費電力をより適切な量に収めることによって，対象ネットワークにおける通信中のリンクの省電力化を実現する．なお，提案方式では，中間帯域を実現するにあたり，ポートの増設は行わず，適用

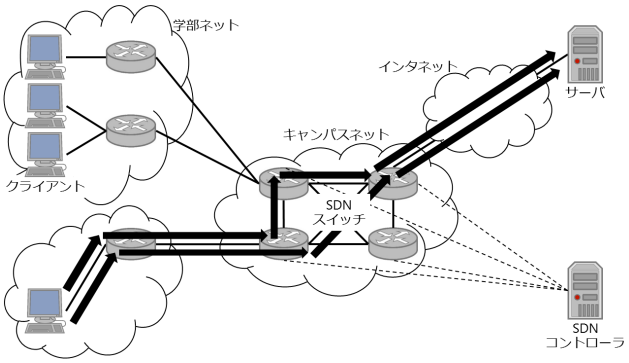


図 1 提案方式で想定するネットワーク
Fig. 1 Target network of the proposed method.

対象のネットワークにおいて実現可能な範囲でマルチパス通信を行うものとする．また，本論文では SDN を実現するにあたり，OpenFlow 1.0 を使用する．

4.2 アルゴリズム

まず，本節で用いる変数などを以下のように定義する．

- i，j，k：任意の値を表す変数
- L_i：スイッチ間の任意のリンク（$i = 0, 1, \ldots$）
- B_i：L_i の帯域
- E_i：L_i の通信時の消費電力
- B_{thresh}：ネットワークが 1 つの MPTCP コネクションに提供することを目標とする最低帯域
- s：各拠点ネットワークが接続されている入口エッジスイッチ
- P_i：入口エッジスイッチ s と外部ネットワークが接続されている出口エッジスイッチ間の排他的なパス（$i = 0, 1, \ldots, n_s - 1$）
- n_s：排他的なパスの本数．1 つの MPTCP コネクションはこれと同数のサブフローを持つ（正確には，クライアントは十分な数のサブフローを生成し，n_s 本を超えるサブフローはコントローラが廃棄する）
- n_p：サブフローに割り当てているパスの数（$n_p \leq n_s$）
- n_c：通信中の MPTCP コネクションの数

提案するアルゴリズムにおいて，スイッチ間の任意のリンク L_i は，低帯域低消費電力状態 Low と高帯域高消費電力状態 High のいずれかにあることを前提とする．そして，L_i の帯域 B_i および通信中の消費電力 E_i をそれぞれ式 (1)，式 (2) と定義し，非通信時の消費電力は，式 (1) の値よりも十分小さいものとする．次に，ネットワークが 1 つの MPTCP コネクションに提供することを目標とする最低帯域を B_{thresh} とする．MPTCP は輻輳制御を行うため，1 つのパスにあまり多くのコネクションを入れると，それぞれが得られる帯域が小さくなり，ユーザの要求に応えられない可能性がある．そこで，目標帯域 B_{thresh} を定めることで，パスに入れるコネクション数を制限する．この目標帯域は，各拠点ネットワークに接続されているクラ

イアントがエッジスイッチ間の最短パス（パス上のリンクは常時 High）を経て外部ネットワークと通信を行うとき，つまり，提案方式を適用しない通常時において，各クライアントが使用している平均的な帯域以下になるようにネットワーク管理者が設定する．

$$B_i = \begin{cases} B_{\text{low}} & (L_i \text{ is at Low.}) \\ B_{\text{high}} & (L_i \text{ is at High.}) \end{cases} \quad (1)$$

$$E_i = \begin{cases} E_{\text{low}} & (L_i \text{ is at Low.}) \\ E_{\text{high}} & (L_i \text{ is at High.}) \end{cases} \quad (2)$$

4.2.1 アルゴリズムの概要

以上の前提のもとで，提案方式であるスイッチ間帯域切替えのアルゴリズムの概要を以下に示す．

(1) コントローラは，スイッチやそのポートの情報を把握する．

(2) コントローラは，さらに，ネットワークトポロジを把握する．

(3) コントローラは，各拠点ネットワークが接続されている入口エッジスイッチ s と外部ネットワークが接続されている出口エッジスイッチ間に排他的パスを求める．

(4) MPTCP コネクションがないときは，すべてのリンクの状態を Low にする．
MPTCP コネクションが新たに追加され，通信中の MPTCP コネクションが n_c 本になったとする．ここで，排他的なパスの本数が n_s のとき，前節での n_s の説明にあるように，各 MPTCP コネクションはちょうど n_s 本のサブフローを持つため，合計（$n_s \times n_c$）本のサブフローが存在することになる．コントローラは，これらのサブフローに対して，目標帯域を満たすのに最低限必要な n_p 本のパスを均等に割り当て，残りの（$n_s - n_p$）本のパスには，サブフローを割り当てず，リンクや存在するならば中継するスイッチの消費電力を可能な限り削減する．
ただし，n_s 本のサブフローでは目標帯域を満たせないときは，一部のパスのリンクの状態を High にして，サブフローに割り当てる．

(5) MPTCP コネクションが切断・終了したとき，残りのコネクションのサブフローへのパスの割当てを整理する．

4.2.2 アルゴリズムの詳細

以下では，アルゴリズムの詳細を説明する．

まず，事前準備として，コントローラは，始点と終点となるスイッチ間の排他的なパスを探索する．

(1) コントローラは，スイッチやポートの新規接続・切断を，スイッチからの Hello メッセージまたは Port Status メッセージから検知すると，ネットワークに接続された各スイッチに対して，スイッチの情報を要求する Features Request メッセージを送る．そして，その応答である Features Reply メッセージから，Datapath ID やポート一覧などの情報を取得する．

(2) コントローラは，(1) で得た情報をもとに，各スイッチに出力ポートを指定した LLDP（Link Layer Discovery Protocol）パケットを Packet Out メッセージとして送り，各スイッチは，受け取ったパケットを指定されたポートから出力する．もし，この出力ポートの先に別のスイッチが接続されていれば，そのスイッチに LLDP パケットが届く．LLDP パケットを受け取ったスイッチは自身のフローテーブルを参照するが，そのパケットを処理するためのフローエントリがインストールされていない．このため，受け取ったパケットに，Datapath ID やパケットを受信したポートの番号などの情報を加えた Packet In メッセージをコントローラに送る．コントローラはこの Packet In メッセージを解析して，LLDP パケットを送らせたスイッチと受け取ったスイッチの間にあるリンクを検出する．そして，これをすべてのスイッチのすべてのポートに対して行うことで，コントローラはネットワーク全体のトポロジを把握し，グラフを作成する．

(3) (2) で作成したグラフから，コントローラは，各拠点ネットワークが接続されているエッジスイッチ s と外部ネットワークが接続されているエッジスイッチ間の排他的なパスを，以下の手順でホップ数が少ない順に求める．なお，本論文では，すべての NIC が同一である前提のため，ここでいうホップ数が少ない順とは，消費電力が低い順となることを示している．NIC が同一ではない場合には別の計算方法が必要となるが，これについては 5 章で考察する．
まず，最短パス P_0 をダイクストラ法で求める．次に，グラフから P_0 を除き，再度ダイクストラ法で P_1 を計算し，以下同様に，式 (3) を満たす間，もしくはエッジスイッチ間のパスがなくなるまで，これを繰り返す．そして，これにより，入口エッジスイッチ s と出口エッジスイッチとの間に n_s 本の排他的なパス $P_0, P_1, \ldots, P_{n_s-1}$ が得られたとする．

$$\sum_{i \geq 0} \sum_{L_j \in P_i} E_j(L_j \text{ is at Low.}) \leq \sum_{L_k \in P_0} E_k(L_k \text{ is at High.}) \quad (3)$$

ここで，$\sum_{L_j \in P_i} E_j\,(L_j \text{ is at S.})$ はパス P_i 上のすべてのリンク L_j が状態 S であり，かつ通信中であるときの，パス P_i の総消費電力を示しており，式 (3) は，通信を行っていないリンクの消費電力が十分小さいとして無視したとき，P_i 上の全リンクが Low である n_s 本のパスを使ったマルチパス通信の総消費電力が，全リンクが High である最短パス P_0 を使った通信のときよりも小さいことを表している．

以上により，各拠点ネットワークごとに，始点と終点となるエッジスイッチ間に排他的なパスが決定された．提案方式では，エッジスイッチ s に接続している拠点ネットワークにあるクライアントがMPTCP通信を開始すると，クライアントとサーバ間に，n_s 本のサブフローによる1つのMPTCPコネクションを構築する．このとき，n_s は拠点ネットワークごとに異なるので，提案方式におけるクライアントは，十分な数のサブフローの追加を要求するものとする．ただし，サブフローの追加要求を示すパケットは，新規のIPアドレスとポート番号の組でクライアントから送られてくるため，そのつどPacket Inが発生し，コントローラに転送される．これに対して，コントローラは n_s 本目までのサブフローの追加要求に対して，以下の手順で排他的なパスを割り当て，それ以降の追加要求に対しては，そのパケットを破棄することで，クラインアントとサーバ間に n_s 本のサブフローによるMPTCPコネクションを張ることができる．したがって，通信中のMPTCPコネクションの数が n_c であったとすると，サブフローの総数は，$(n_s \times n_c)$ 本になる．コントローラは，これらのサブフローにいずれかのパスを割り当てる必要がある．

ここで，パスの割当てとは，サブフローに割り当てるパス上のスイッチに，Flow Modメッセージによってフローエントリをインストールすることを意味する．Flow Modメッセージには，そのフローエントリのタイムアウト時間を設定できるが，提案方式では，エントリが最後に参照されてから消去されるまでの時間を指定するIdle Timeoutを設定する．そして，このタイムアウトによってフローエントリが消えたとき，スイッチからそれを知らせるFlow Removedメッセージが届くので，その到着をもってコネクションの切断・終了を判断する．

(4)【A. 新規MPTCPコネクションの検出】

サブフローの始点となるエッジスイッチ s は，フローエントリがインストールされていないMPTCPパケットを受け取ると，これをPacket Inメッセージに格納して，コントローラに送る．コントローラは，このPacket Inメッセージを解析して，送信元・送信先のIPアドレスを取得する．

【B. 新規MPTCPコネクションのサブフローへのパスの割当て】

もし，そのIPアドレスの組が新規のものであればパスの割当てを行うが，このとき，コントローラは，(3)で求めたパスセットより，P_0 から順にパスを取り出して以下の手順で割り当てる．

【B.1 通信中のコネクションがない場合】

まず，通信中のコネクションがない場合は，パスの割当ては行わず，コントローラは，(2)で把握したトポロジをもとに，すべてのリンクの状態をLowに切り替える．なお，OpenFlowにはリンクの状態を変更するメッセージがないので，他のプロトコルを使用する必要がある．提案方式の通信実験では，コントローラが各スイッチにSSHで接続し，ifconfigやethtoolなどのコマンドを実行して，これを実現している．

【B.2 Low状態の n_p 本のパスが割当て済みであり，新規接続により，コネクション数が n_c となった場合】

次に，単一の拠点ネットワークに接続する単一もしくは複数のクライアントからのコネクションのみが存在し，それらのためにLow状態の n_p 本（$0 < n_p < n_s$）のパス $P_0, P_1, \ldots, P_{n_p-1}$ が割り当てられているとする．このとき，これらのパスで提供できる総帯域は $n_p \times B_{\text{low}}$ である．新規接続により，通信中のMPTCPコネクションの数が n_c 本となったとすると，これらのコネクションの目標最低帯域の総和は $n_c \times B_{\text{thresh}}$ である．このとき，式(4)を満たすかどうかで，コントローラの挙動が変化する．

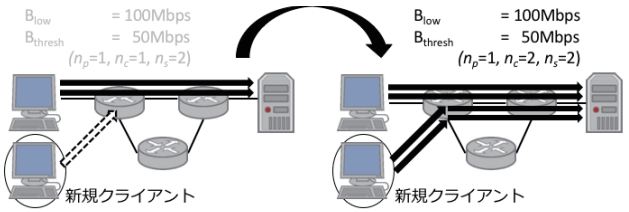
$$n_p \times B_{\text{low}} \ \geq \ n_c \times B_{\text{thresh}} \tag{4}$$

なお，B_{low} と B_{thresh} は事前に与えられており，また，コントローラはサブフローに割り当てているパスの本数 n_p を管理しているので，【A】の新規コネクションの検出による n_c の変動に合わせて，コントローラは式(4)を容易に計算できる．

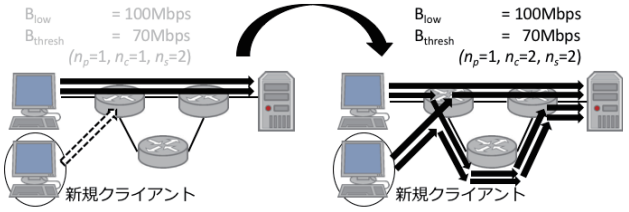
(a)**【B.2a 追加されたコネクションが，割当て済みのパスに収容可能な場合】**

式(4)を満たす場合は，現在使用中のパスに新規接続によるコネクションを加えても，目標最低帯域を各コネクションに提供できる．このため，コントローラは，使用中の n_p 本のパス $(P_0, P_1, \ldots, P_{n_p-1})$ に対して，割当て済みのサブフロー数の少ない順に，最小のものが複数ある場合はパスの添え字の若い順に，新規接続のコネクションが持つ n_s 本のサブフローを割り当てるよう，パス上の各スイッチにフローエントリをインストールする．

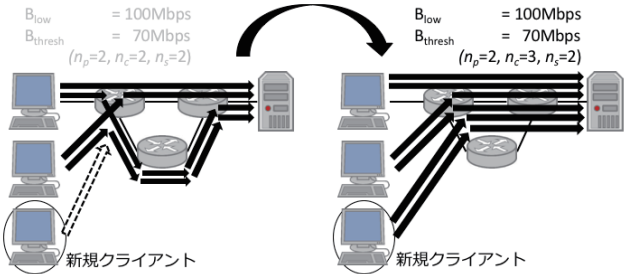
たとえば，**図2**(a)では，各リンクのLow状態の帯域 B_{low} が100 Mbps，目標最低帯域 B_{thresh} が50 Mbpsであり，使用可能なパスの本数 n_s が2本であるうち，現在，サブフローに割り当てられているパスの本数 n_p が1本である（同図左）．このとき，新規接続による2つ目のMPTCPコネクションを加えてコネクションの数 n_c が2となったとしても，目標最低帯域の総和100 Mbps $(= n_c \times B_{\text{thresh}})$ は1本のパスで提供できる帯域100 Mbps $(= n_p \times B_{\text{low}})$ 以下となり，式(4)を満たす．以上のことから，1本目のパス P_0 に収容できることが分かったので，コントローラは，新規コネクションが持つ n_p $(= 2)$ 本のサブフローを，



(a) 目標最低帯域を満たせる場合



(b) 目標最低帯域を満たせず，割当て可能なパスがある場合



(c) 目標最低帯域を満たせず，割当て可能なパスがない場合

図 2 提案方式におけるサブフローのパス割当て
Fig. 2 Path assignments for subflows in the proposed method.

パスを増やさずそのまま収容している（同図右）．

(b)【B.2b 追加されたコネクションが，割当て済みのパスに収容できない場合】
式 (4) を満たさない場合は，現在使用中のパスに新規接続によるコネクションを加えると，目標最低帯域を各コネクションに提供できない．

【B.2b-1 パスが追加できる場合（$n_p < n_s$）】
このため，コントローラは新しいパス P_{n_p} を取り出し，$(n_s \times n_c)$ 本のサブフローが $(n_p + 1)$ 本のパスへ均等に乗るように，各パス上のスイッチにフローエントリをインストールし直す．
たとえば，図 2 (b) において，現在のパスの割当て状況（同図左）は，図 2 (a) 左と同じであるが（$n_p = 1$，$n_s = 2$），$B_{\text{thresh}} = 70$ Mbps であるため，新規接続による 2 つ目のコネクションを加えると（$n_c = 2$），目標最低帯域の総和 140 Mbps（$= n_c \times B_{\text{thresh}}$）は P_0 に収容できない．そこで，P_1 を加えて（$n_p = 2$），合計 200 Mbps（$= n_p \times B_{\text{low}}$）とし，2 つのコネクションが持つ合計 4 本のサブフローを，P_0 と P_1 に均等に割り振っている（同図右）．このように，最短パスの状態を High に切り替え，過剰な帯域を高い消費電力で使用するのではなく，2 本のパスによる中間帯域を導入することで，最低限の帯域を低い消費電力で提供することができる．

【B.2b-2 パスが追加できない場合（$n_p = n_s$）】
ただし，$n_p = n_s$ であり，新たなパスが追加できない場合は，中間帯域の終了となる．たとえば，図 2 (c) 左のように，通信中の 2 つのコネクションのサブフローにパスセットの全パスが割り当てられており（$n_p = n_s = 2$），$B_{\text{thresh}} = 70$ Mbps の状態で，新たな 3 つ目のコネクションが加わると（$n_c = 3$），目標最低帯域の総和 210 Mbps（$= n_c \times B_{\text{thresh}}$）は 2 本のパスでも収容できない．このため，コントローラは，各コネクションに割り当てているパスをすべて P_0 に集約するため，P_0 上のスイッチにすべてのフローエントリをインストールするとともに，他のパスのフローエントリを削除する．そして，P_0 上のスイッチに接続し，リンクの状態をすべて High に切り替える（同図右）．
この後，コネクションがさらに追加され，全リンクが High 状態となった P_0 でも収容できなくなった場合には，グラフから P_0 を除いたうえで，アルゴリズムを (3) から再び適用する．これにより，P_0 以外の，全リンクが Low 状態のパスを 1 本ずつ追加していき，それでも収容できなくなった場合には，そのうちの 1 本を，全リンクの状態を High にして収容する．また，これを再帰的に繰り返すことで，提案方式では，(排他的なパス数) × (High 状態の帯域)/B_{thresh} 本分のコネクションを収容することが可能である．

【C. 複数の拠点ネットワークから MPTCP コネクションがあるとき】
接続先の拠点ネットワークが異なる複数のクラアントが存在する場合も以上と同様の処理を行うが，他の拠点ネットワークからのパスも含めて，リンク単位で式 (4) を確認する点が異なる．このとき，High 状態のリンクを複数のサブフローで共有することで，効率の良いパスを構成することが可能になると考えられるが，リンクの状態の組合せは，パス中のリンク数にともなって指数関数的に増加するため，各拠点ネットワークの入口エッジスイッチから出口エッジスイッチの間のホップ数が大きいネットワークでは，最適な組合せを実用的な時間で得ることは困難であると考えられる．さらに，High 状態のリンクを複数のサブフローで共有することで，式 (4) を満たす排他的なパスの本数 n_s が変動することも考えられる．

(5)【D. MPTCP コネクションの切断・終了時】
スイッチから Flow Removed メッセージが届くと，コントローラはそのメッセージを解析して，消えたフローエントリの対象としていたコネクションが切断・

終了したと判断し（一時的に中断してはいたが切断・終了はしていなかった場合には，通信が回復した後，Packet In が発生し，改めてパスの割当てが行われる），サブフローへのパス割当てを (4) の手順で整理する．もし，式 (4) を満たす状況に戻った場合は，再び中間帯域に移行する．このとき，High 状態のパスが 1 本の場合，P_0 のリンクの状態はすべて High になっているため，コントローラはパスの割当て後，P_0 上のスイッチに接続して Low に切り替える．複数の High 状態のパスを使用している場合は，パスの添字が最も大きい，つまり使用している High 状態のパスの中で最も消費電力の高いパスに同様の処理を行う．

なお，リンクの状態を切り替える処理において，通信中断が発生する恐れがある．この改善には，2.3.2 項で説明した著書らの従来研究での手法を導入することが選択肢としてあげられる．具体的には，アルゴリズムの【B.2b-2】において，中間帯域から高帯域に切り替える際，コントローラは，すべてのサブフローを最短パスの P_0 に集約する前に，一時的に，最短パスの次に短いパス P_1 に集約する．そして，P_0 上のすべてのリンクの状態が High へ切り替わった後に，P_1 に集約したサブフローをすべて P_0 に移動する．これにより，一時的に帯域が低くなるものの，帯域切替え時間をほぼ 0 にすることができると考えられる．

ここで，すべてのサブフローを移動させるのではなく，P_0 で通信を行っているサブフローのみを P_1 に移動することも考えられる．しかし，移動しなかった P_0 と P_1 以外に割り当てられたサブフローはこの影響を受けず，帯域の公平性の観点から望ましくない．このような公平性は無視し，システム全体のスループットを重視することも考えられるが，MPTCP コネクションはすべてのサブフローに分散してパケットを送信しているため，一部のサブフローへの影響が，コネクション全体に影響が及ぶことが考えられる．このため，本アルゴリズムでは，すべてのサブフローを P_1 に移動している．なお，前述のとおり，OpenFlow ではリンク状態を切り替えられないため，提案方式ではこのような手順をとったが，OpenFlow のみでリンク状態が切り替えられ，サブフローを P_1 に移動することなく，切替え後のパスにアトミックに切り替えることができれば，切替え時間をさらに短縮できると考えられる．

以上が提案方式のアルゴリズムになるが，リンクの総数を n としたとき，その計算量は次のようになる．まず，アルゴリズムのステップ (1) から (3) はネットワーク構成が変わらない限り，開始時に 1 度実施するだけでよい．ネットワーク構成が変わると，ステップ (1) と (2) は差分を増減するだけでよいので，ネットワーク構成をハッシュテーブルに格納するなどすれば $O(\log n)$ である．ステップ (3) では排他的なパスを求めるために，ダイクストラ法を n_s 回実行するので $O(n^2)$ である．なお，n_s を求めるため，排他的なパスが見つかるたびに式 (3) を実行する必要があるが，右辺は最短経路のリンクの状態がすべて High のときの消費電力の総和，左辺は見つかった排他的なパスのリンクの状態がすべて Low のときの消費電力の総和の累積値なので，各々の計算は，$O(n)$ である．

ステップ (4) において，【A】の新規コネクションの検出は，コネクション数 n_c に依存するが，IP アドレスとポート番号の組でハッシュテーブルに格納するなどすれば，$O(\log n_c)$ の検索で済む．【B】では，【B.2】が主な処理になるが，式 (4) の計算は【B.2】で述べたとおり容易で，$O(1)$ である．【B.2a】は n_s 本のサブフローを割り当てる処理が必要なので，$O(n)$ である．【B.2b】は $(n_s \times n_c)$ 本のサブフローを割り当てる処理が必要なので，$O(n \times n_c)$ である．【C】では，拠点ネットワークの数だけ主に【B】の処理を行うことになるが，拠点数は n よりも十分小さいと考えると，拠点数の増加は，計算量の増加にそれほど寄与しない．ただし，本論文の対象外となるが，出口エッジスイッチが複数あった場合は，その処理は $O(n^2)$ となることが予想され，無視できないと考えられる．最後に，ステップ (5) は【A】と同様に，$O(\log n_c)$ である．

以上のことから，ネットワーク構成がほとんど変更されず，リンク使用率が低くて，n_c よりも n の方が十分大きい場合は，【B.2b】の処理は少なくなり，【B.2a】の $O(n)$ が支配的になるので，本アルゴリズムは，n が大きいネットワークでも対応できる．その逆に，n よりも n_c の方が十分大きい場合は，$O(n \times n_c)$ が支配的になるので，割り当てるパス本数を増やす【B.2b】の処理を可能な限り抑えることが重要であると考えられる．たとえば，$B_{\mathrm{low}} = 100\,\mathrm{Mbps}$，$B_{\mathrm{thresh}} = 5\,\mathrm{Mbps}$ として，Low 状態の排他的なパス 1 本に $B_{\mathrm{low}}/B_{\mathrm{thresh}} = 20$ 本までのコネクションを収容できるようにし，最も処理のかかる【B.2b】を 20 コネクションに 1 回程度，それ以外のときは，処理の軽い【B.2a】で済ませるようにする．これにより，パスあたりの平均ホップ数が 3，排他的なパスの本数が 10 本であるような，リンク総数 n = (平均 3 ホップ) × (10 パス) = 30 本で構成されるネットワークにおいて，数拠点程度（2 パスあたりに 1 拠点と見積もり）から同時に 200 コネクションまで，割り当てるパスの本数を増やさずに収容することができると考えられる．

4.3 評価

本節では，提案方式の有用性を確認するために行った通信実験について説明する．本実験のために構築したネットワークを**図 3** に，使用した機器を**表 1** に示す．

機材の都合から，クライアントとサーバは，1 台の PC に 3 枚の NIC を搭載して，それぞれに異なるサブネットを与えることで実現した．OS のカーネルには，MPTCP Linux カーネル実装 v.0.89 [21] を利用し，クライアントが

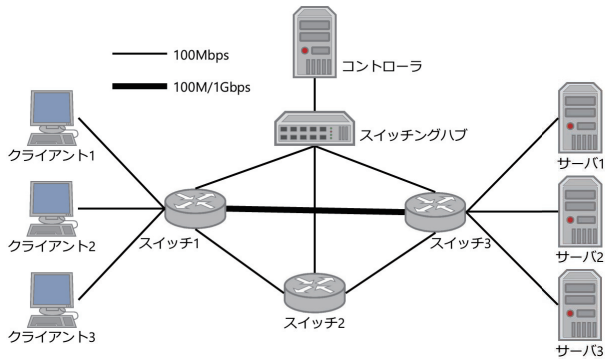


図 3 通信実験のネットワーク構成
Fig. 3 Network topology for the communication experiments.

追加を要求するサブフロー数は 10 とした．サブフロー追加要求の送信間隔は，カーネル実装のデフォルトのままで，サブフローにパスが割り当てられているときの送信間隔は 1 m 秒未満だが，追加要求が破棄されると，送信間隔は指数的に長くなっていく．スイッチは，NIC を必要枚数搭載した PC に Open vSwtich v.2.6.0 [27] を導入することで実現し，スイッチングハブを介してコントローラと接続する．コントローラには，Trema v.0.9.0 [28] を使用している．各機器間のリンク帯域は 100 Mbps とし，最短パス上のスイッチであるスイッチ 1 とスイッチ 3 の間のリンク帯域は，$B_{\mathrm{low}} = 100\,\mathrm{Mbps}$，$B_{\mathrm{high}} = 1\,\mathrm{Gbps}$ とした．また，目標最低帯域は $B_{\mathrm{thresh}} = 100\,\mathrm{Mbps}$ に設定した．リンクの消費電力に関しては，表の環境における 100BASE-TX と 1000BASE-T の電力を事前に計測した結果，E_{low} と E_{high} の比は 1：3 となった．したがって，中間帯域では低帯域リンクからなるパスを 2 本まで使用でき，2 本でまかなえなくなったときは，高帯域へと切り替わる．

以上の環境において，図 3 におけるクライアント 1 からサーバ 1，クライアント 2 からサーバ 2，そしてクライアント 3 からサーバ 3 へ，それぞれ iperf を用いた MPTCP による通信を，**図 4** に示すトラフィックパターンに沿うように隙間なく行った．この図における 2 つのトラフィックは，提案方式が対象とするネットワークを想定した以下のトラフィックパターンを模したものである．

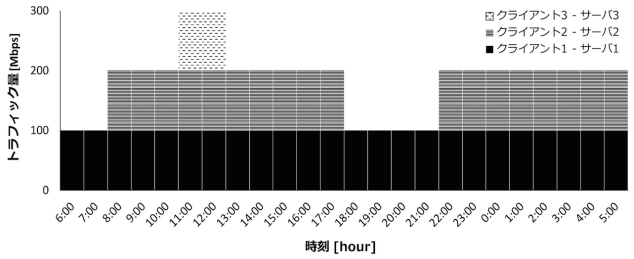
(a) キャンパスネットワーク 日中授業が行われているときにはトラフィックが増大し，正午付近でピークに達する．また，定期バックアップなどの影響で深夜にも再びトラフィックが増える．

(b) ISP 間ネットワーク 通勤時や始業直後にトラフィックが一時的に増大する．午後から企業活動などによってトラフィックが増え始め，終業後にピークを迎える．

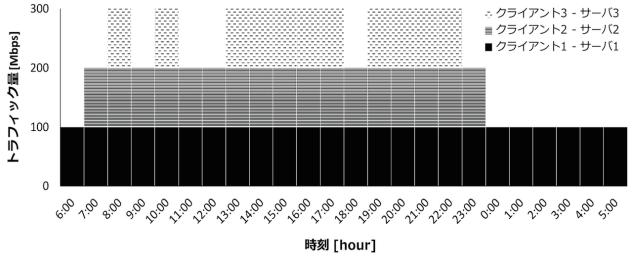
また，実際のネットワーク環境では定常的なトラフィックがあると考えられることから，MPTCP による通信だけではなく，クライアント 2 からサーバ 2，クライアント 3 からサーバ 3 へ，それぞれ 10 Mbps（データサイズ 64 バイト，パケット長 92 バイト）の UDP による CBR トラフィックも同時に流す．なお，各スイッチには，クライアント 2 からサーバ 2 への CBR トラフィックは最短パスを，クライアント 3 からサーバ 3 への CBR トラフィックはスイッチ 2 を経由するパスを通るように，あらかじめフロー

表 1 使用機器仕様表
Table 1 Equipment specifications list for the communication experiments.

クライアント	機種	PRECISION T1500, Dell
	CPU	Core i7-860 2.8 GHz, Intel
	OS	Ubuntu 14.04 LTS
	メモリ	8 GB
	NIC（クライアント 1）	57780 Gigabit Ethernet Controller, Broadcom
	NIC（クライアント 2）	LGY-PCI-TXD, Baffalo
	NIC（クライアント 3）	LGY-PCI-TXD, Baffalo
サーバ	機種	PRECISION T1500, Dell
	CPU	Core i7-860 2.8 GHz, Intel
	OS	Ubuntu 14.04 LTS
	メモリ	8 GB
	NIC（サーバ 1）	57780 Gigabit Ethernet Controller, Broadcom
	NIC（サーバ 2）	LGY-PCI-TXD, Baffalo
	NIC（サーバ 3）	LGY-PCI-TXD, Baffalo
スイッチ 1	機種	Express5800/GT110b, NEC
	CPU	Celeron G1101 2.26 GHz, Intel
	OS	Ubuntu 14.04 LTS
	メモリ	4 GB
	NIC（クライアント 1 接続）	Intel Pro/1000 PT D33682 Dual Port Server Adapter, Intel
	NIC（クライアント 2 接続）	Intel Pro/1000 PT D33682 Dual Port Server Adapter, Intel
	NIC（クライアント 3 接続）	UE-200TX-G, PLANEX
	NIC（スイッチ 2 接続）	LGY-PCI-TXD, Baffalo
	NIC（スイッチ 3 接続）	82574L Gigabit Ethernet Controller, Intel
	NIC（コントローラ接続）	UE-200TX-G, PLANEX
スイッチ 2	機種	PRECISION T1500, Dell
	CPU	Core i3-540 3.06 GHz, Intel
	OS	Ubuntu 14.04 LTS
	メモリ	4 GB
	NIC（スイッチ 1 接続）	LGY-PCI-TXD, Baffalo
	NIC（スイッチ 3 接続）	LGY-PCI-TXD, Baffalo
	NIC（コントローラ接続）	UE-200TX-G, PLANEX
スイッチ 3	機種	Express5800/GT110b, NEC
	CPU	Celeron G1101 2.26 GHz, Intel
	OS	Ubuntu 14.04 LTS
	メモリ	4 GB
	NIC（サーバ 1 接続）	Intel Pro/1000 PT D33682 Dual Port Server Adapter, Intel
	NIC（サーバ 2 接続）	Intel Pro/1000 PT D33682 Dual Port Server Adapter, Intel
	NIC（サーバ 3 接続）	UE-200TX-G, PLANEX
	NIC（スイッチ 1 接続）	82574L Gigabit Ethernet Controller, Intel
	NIC（スイッチ 2 接続）	LGY-PCI-TXD, Baffalo
	NIC（コントローラ接続）	UE-200TX-G, PLANEX
コントローラ	機種	Magnate JD, Diginnos
	CPU	Core i5-3470 3.20 GHz, Intel
	OS	Ubuntu 14.04 LTS
	メモリ	8 GB
	NIC（各スイッチと接続）	RTL8111E Gigabit Ethernet Controller, Realtek
電力計	機種	TAP-TSTUP121, SANWA SUPPLY



(a) キャンパスネットワークを想定したトラフィックパターン



(b) ISP 間ネットワークを想定したトラフィックパターン

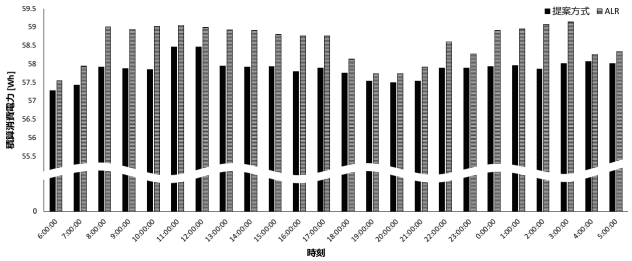
図 4 通信実験で用いたトラフィックフロー

Fig. 4 Network traffic flows of the communication experiments.

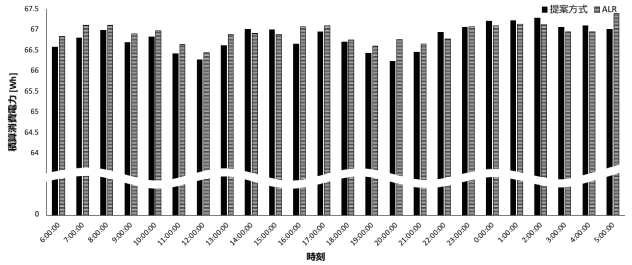
エントリをインストールしている．

これらのトラフィックのもとで，3 台のスイッチの 24 時間分の積算消費電力と各クライアントとサーバ間のスループットおよび通信中断時間を計測した．さらに，従来の ALR でも同様の実験を行い，提案方式との比較を行った．ここで，従来の ALR とは，前節のアルゴリズムで $n = 1$ に固定し，使用するパスを最短パス P_0 のみに限定することで，中間帯域を省いたものである．また，実験では，前節で述べた著者らの従来研究の導入は行っていない．

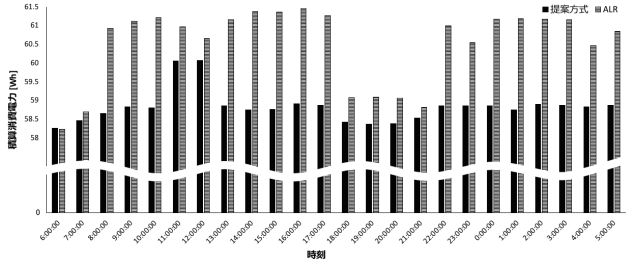
図 5 に，トラフィックパターン (a) におけるスイッチ別の 1 時間あたりの積算消費電力を示す．**図 6** には，各スイッチの合計消費電力を示す．これらの図から，提案方式は従来の ALR よりも，3 台のスイッチの合計消費電力が，24 時間で 59.9 Wh（電力比では 1.35%）低くなることが分かった．特に，トラフィック量が 200 Mbps となる時間帯（図 4 (a) より 8 時〜10 時，13 時〜17 時，22 時〜5 時）に消費電力が少なくなり，スイッチ 1 では平均で 0.87 Wh（電力比では 1.48%），スイッチ 2 では平均で 0.05 Wh（電力比では 0.07%），スイッチ 3 では平均で 2.26 Wh（電力比では 3.70%）低いことが分かる．これらの時間帯において，提案方式では中間帯域，ALR では高帯域を用いており，100 Mbps のリンクによる 2 本のパスを用いた方が，1 Gbps のリンクによる 1 本のパスを用いるよりも消費電力が小さくなるためである．また，スイッチ 2 は，CBR トラフィックが絶えず流れているため，2.1 節で述べた Ethernet の電力特性のとおり，MPTCP トラフィックが流れているかどうかは消費電力にほぼ影響がない．そのため，図 5 (b) のいずれの区間においても，提案方式と ALR の消費電力に有意な差は見当たらない．以上より，提案方式を適用するこ



(a) スイッチ 1 の 1 時間あたりの積算消費電力



(b) スイッチ 2 の 1 時間あたりの積算消費電力



(c) スイッチ 3 の 1 時間あたりの積算消費電力

図 5 図 4 (a) におけるスイッチ別の消費電力

Fig. 5 Power consumption of each switch in the case of Fig. 4 (a).

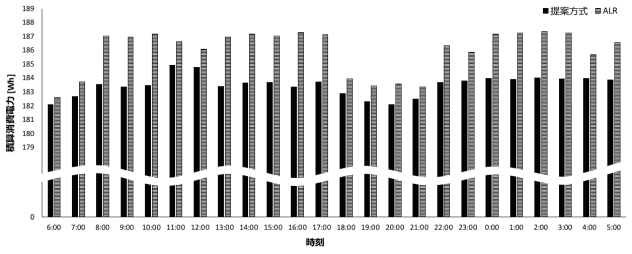
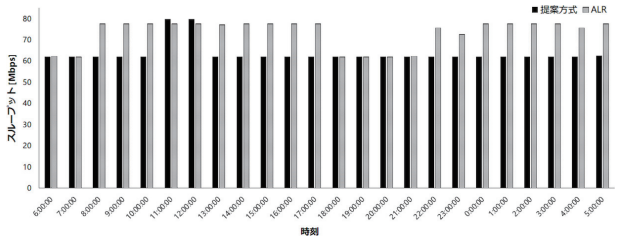


図 6 図 4 (a) における各スイッチの合計消費電力

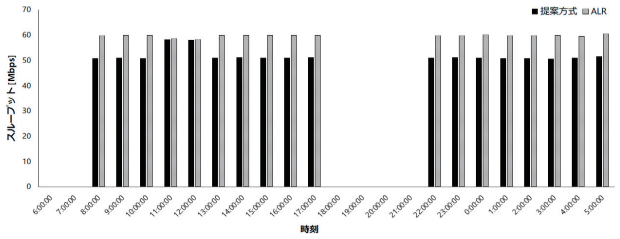
Fig. 6 Total power consumption of each switch in the case of Fig. 4 (a).

とで，図 4 (a) のようなトラフィックにおいて，OpenFlow ネットワークを構成するスイッチの省電力化を実現できることが分かった．

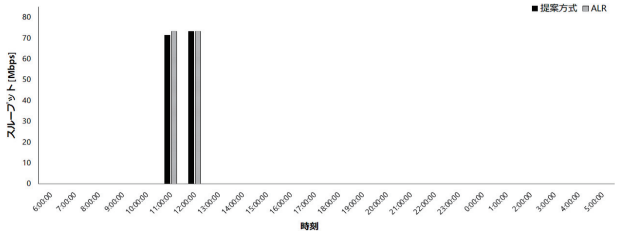
図 7 に，トラフィックパターン (a) における，各クライアントとサーバ間の 1 時間あたりの平均スループットを示す．この値は MPTCP のみのスループットであり，スループットの計算に CBR は含まれていない．図 7 (a) と図 7 (b) より，提案方式が中間帯域を提供している時間帯における，提案方式の平均スループットは，クライアント 1 からサーバ 1 への通信で ALR のときの 80.0%，クライアント 2 からサーバ 2 への通信では 84.9%にとどまってい



(a) クライアント 1 とサーバ 1 間の 1 時間あたりの平均スループット



(b) クライアント 2 とサーバ 2 間の 1 時間あたりの平均スループット

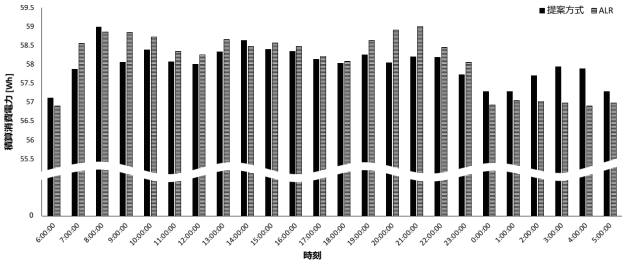


(c) クライアント 3 とサーバ 3 間の 1 時間あたりの平均スループット

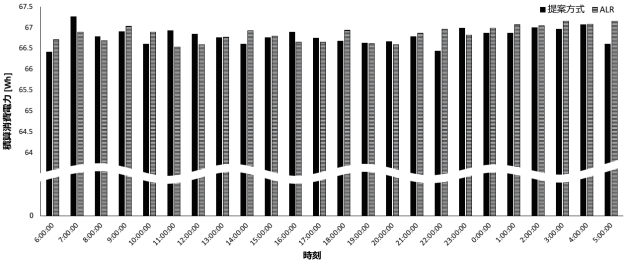
図 7 図 4 (a) におけるクライアントとサーバ間のスループット
Fig. 7 Throughput of each connection in the case of Fig. 4 (a).

る．この原因として，Linux カーネル実装の MPTCP では，各サブフローへのパケットスケジューリングの重みに RTT 比を用いていることがあげられる．つまり，パス P_0（スイッチ 1–スイッチ 3）に比べて経路長の長い P_1（スイッチ 1–スイッチ 2–スイッチ 3）を通るサブフローのスループットは，P_0 を通るものよりも小さくなる．一方で，スケジューリングの重みが大きい P_0 を通るサブフローも，2 つのコネクションが同時に P_0 で通信を行っているため，スループットがそれほど大きくならず，結果としてコネクション全体のスループットが小さくなったと考えられる．このことから，MPTCP のパケットスケジューリングを変更し，すべてのサブフローに均等にパケットを送るようにすれば，スループットが改善されると考えられる．たとえば，Linux カーネル実装では，RTT 比以外にも，ラウンドロビンによるスケジューリングが可能になっている．ただし，今回の実験では目標最低帯域を 100 Mbps としたため，2 本の 100 Mbps のパスで 2 つのコネクションが通信可能であったが，目標最低帯域がこれよりも小さく，より多くのコネクションが通信できるときは，スループットの大きなパスを通るサブフローにパケットがより集中してしまい，スループットがさらに低下するおそれがある．

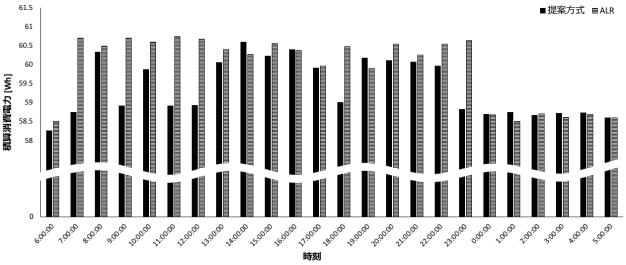
次に，**図 8** に，トラフィックパターン (b) におけるスイッチ別の 1 時間あたりの積算消費電力を，**図 9** に各スイッチの合計消費電力を示す．これらの図から，提案方式は従来の ALR よりも，3 台のスイッチの合計消費電力が，24 時間で 15.73 Wh（電力比では 0.35%）低くなることが分かった．提案方式は，特に，提案方式が中間帯域を用いる時間帯（図 4 (b) より 7 時，9 時，11 時～12 時，18 時，23 時）に，ALR よりも，スイッチ 1 は平均で消費電力が 0.40 Wh（電力比では 0.67%），スイッチ 3 は平均で 1.77 Wh（電力比では 2.91%）低いことが分かる．ただし，スイッチ 2 に関しては平均で 0.13 Wh（電力比では 0.20%）高くなっている．図 4 (a) のトラフィックの場合と比べると削減した消費電力量が小さいが，これは，図 4 (b) のトラフィックでは提案方式が中間帯域を用いる時間帯が少な



(a) スイッチ 1 の 1 時間あたりの積算消費電力



(b) スイッチ 2 の 1 時間あたりの積算消費電力



(c) スイッチ 3 の 1 時間あたりの積算消費電力

図 8 図 4 (b) におけるスイッチ別の消費電力
Fig. 8 Power consumption of each switch in the case of Fig. 4 (b).

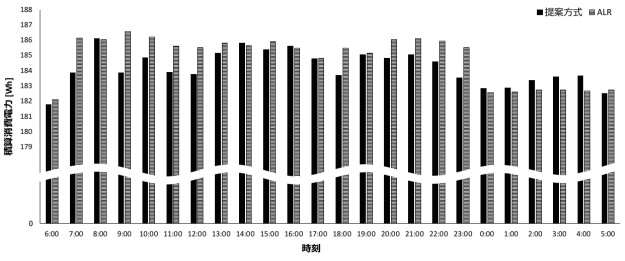
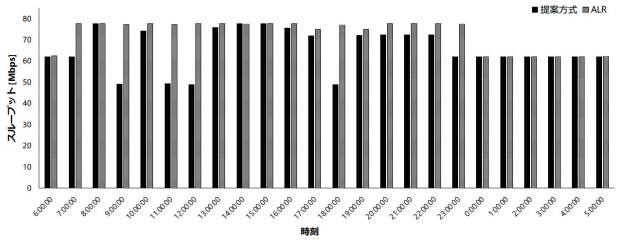
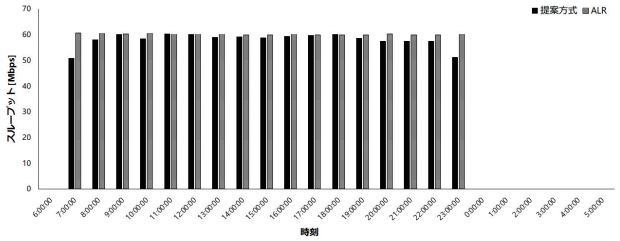


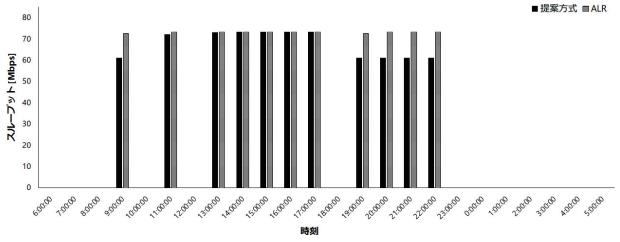
図 9 図 4 (b) における各スイッチの合計消費電力
Fig. 9 Total power consumption of each switch in the case of Fig. 4 (b).



(a) クライアント 1 とサーバ 1 間の 1 時間あたりの平均スループット



(b) クライアント 2 とサーバ 2 間の 1 時間あたりの平均スループット



(c) クライアント 3 とサーバ 3 間の 1 時間あたりの平均スループット

図 10 図 4 (b) におけるクライアントとサーバ間のスループット
Fig. 10 Throughput of each connection in the case of Fig. 4 (b).

く，それ以外の時間帯では ALR と変わらない電力を消費したためである．したがって，提案方式がより有効に働くには，中間帯域でより多くの通信をまかなえるような，それほどトラフィック量の大きくないネットワークを対象とすべきであることが確認できた．

図 10 に，トラフィックパターン (b) における，各クライアントとサーバ間の 1 時間あたりの平均スループット（MPTCP のみ）を示す．図から，トラフィックパターン (a) の場合と同様に，提案方式が中間帯域を提供している時間帯における，提案方式の平均スループットは，クライアント 1 からサーバ 1 への通信で ALR のときの 70.0%，クライアント 2 からサーバ 2 への通信では 93.8%になっている．また，図 7 (a) と比べると，後者の減少は小さくなっているものの，前者ではより減少していることが分かる．これは，高帯域から中間帯域へ移行するために，$P0$ 上に集約していたサブフローを再び $P0$ と $P1$ に分配した際に，クライアント 2 からサーバ 2 への通信が，クライアント 1 からサーバ 1 への通信よりも先に，経路長の短い P_0 でのサブフローの輻輳ウィンドウを広げてしまい，それを長期間維持したことが原因だと考えられる．(b) のトラフィックでは，中間帯域と高帯域の切替えが多いことからも，サブフローの分配にともなうフローエントリの再インストール後に，スループットの大きなパスを誰が先に占有するかがスループットに影響を及ぼしている可能性が高い．このような不公平を改善する方法としては，やはり，MPTCP のサブフローへのパケットスケジューリングの方法を変更することが考えられる．特に，本実験のネットワークのように，選択できるパスの RTT やパケット損失率の差が小さい場合，各コネクションの公平性を考えると，ラウンドロビンによるサブフローへのパケットスケジューリングは有効である．しかし，その差が大きい場合は，サブフロー間でパケットの到着順序が入れ替わるなどの問題が発生し，MPTCP のスループットが減少する可能性があるため，他のパケットスケジューリングを検討する必要がある [29]．

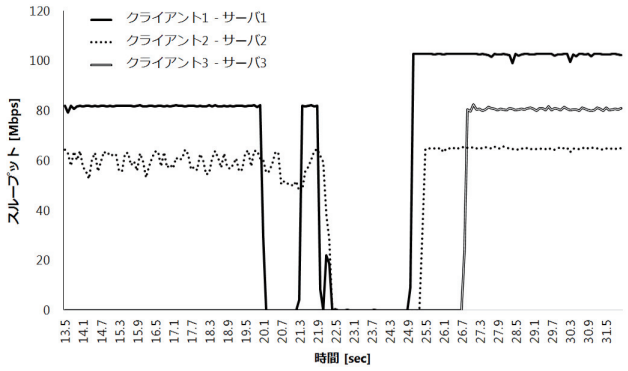


図 11 提案方式における帯域切替え時の通信中断時間
Fig. 11 Communication interruption time by switching the link rate in the proposed method.

最後に，中間帯域から高帯域に切り替わる際の通信状況を見るため，帯域切替え時の各コネクションのスループットを Wireshark で計測した．**図 11** にその結果を示す．図において，実線はクライアント 1 とサーバ 1 間，破線はクライアント 2 とサーバ 2 間，そして二重線はクライアント 3 とサーバ 3 間のスループットをそれぞれ示している．

図において，クライアント 1 とサーバ 1 およびクライアント 2 とサーバ 2 の通信が行われている中，時刻 20.0 秒に，クライアント 3 がサーバ 3 へ通信を開始している．しかし，クライアント 3 からの通信が始まる直前に，他の 2 つのコネクションは 2.5 秒ほど（時刻 22.4 秒～24.9 秒）スループットがゼロになり，通信が中断している．これは，クライアント 3 からの通信が加わったことで中間帯域が終了し，各コネクションは P_0 上に集約されたが，集約後に P_0 上のリンクが High に切り替わり，それにともなう通信中断によって通信が一時的に行えなくなったためである．なお，クライアント 1 からサーバ 1 への通信では，帯域切替えによる通信中断の前（時刻 20.1 秒～21.2 秒）にも通信が中断しているが，これは P_0 に集約される際のフローエントリの再インストールによるものである．これは OpenFlow 特有の問題であり，本論文の対象の範囲外であるので，この対策については今後の課題とする．

前者の通信中断に対し，前節で示した，著書らの従来研究での手法を導入した場合の通信中断時間を**図 12** に示す．

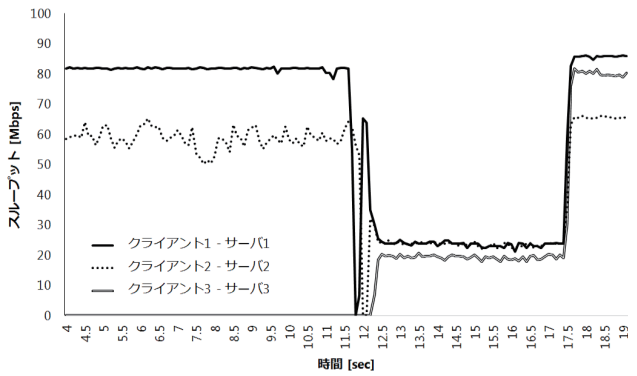


図 12 従来研究での方式を導入した提案方式における帯域切替え時の通信中断時間

Fig. 12 Communication interruption time by switching the link rate in the proposed method adopting our conventional method.

図において，実線はクライアント 1 とサーバ 1 間，破線はクライアント 2 とサーバ 2 間，そして二重線はクライアント 3 とサーバ 3 間のスループットをそれぞれ示している．

図から分かるように，P_0 上のリンクの帯域切替えが完了するまで，P_1 上で通信が行われるため，通信が中断していない．さらに，クライアント 3 とサーバ 3 間の通信は，図 11 では時刻 20.0 秒に開始され，実際のパケットが送られるまでに 6.7 秒かかっているが，図 12 では，時刻 11.7 秒に開始され，その 0.6 秒後にパケットが送られていることから，一時的に P_1 を使うことによって，導入前よりも早く通信を開始することが可能になっている．なお，P_1 を使って 3 つのコネクションを送るため，各スループットは一時的に 1/3 程度に低下するものの，この状況は，P_0 上のリンクが High に切り替わり，通信可能になるまでの 5 秒間ほど（時刻 12.3 秒〜17.5 秒）で解消される．このことから，著書らの従来研究での手法の導入は，提案方式における帯域切替えにともなう通信中断を実質的に防ぐために有効な選択肢であるといえる．

5. まとめ

本論文では，広域ネットワークにおいて，通信中のリンクの省電力化手法を実現することを目的とし，キャンパスネットワークや社内ネットワーク，ISP 間ネットワークといった，十台〜数十台程度のスイッチが接続された中〜大規模の平均リンク使用率が低いネットワークで，すべてのルータが SDN に対応したスイッチで構成されており，それらが単一のコントローラによって制御されるものを対象に，省電力化のための MPTCP による SDN を用いたスイッチ間帯域切替え方式を提案した．そして，通信実験により，通信中断時間やスループット，消費電力を ALR と比較した結果，提案方式はスループットが最大 36.6%低下したものの，各スイッチの積算消費電力を 1 日あたり合計で最大約 60 Wh（電力比では 1.35%）削減できることが分かった．さらに，著者らの従来研究での方式をアルゴリズムに導入することによって，ALR の問題点であった通信中断を防ぐことができることも示した．

今後の課題としては，より一般的な状況において提案方式を適用するために，現在は各拠点につき始点と終点となるエッジスイッチの組合せは 1 組に限定しているが，それらの組合せが複数となる場合や各スイッチの NIC が同一のものではない場合のパスの計算方法について考える必要がある．後者の課題に対しては，現在のパス計算のアルゴリズムに対して，以下 2 点の変更を加えることで対応が可能である．

- ネットワーク管理者は，各 NIC の Low と High における消費電力をそれぞれ事前に計測し，コントローラにおいて，それぞれの計測値と NIC の MAC アドレスとを紐付けたテーブルを作成する．
- コントローラは，ダイクストラによってパスセットを求める際に，上記のテーブルを参照し，グラフにおける各リンクの重み（コスト）をリンクの両端にあたる NIC の消費電力の和に設定する（パスセットはすべてのリンクが High であるときと Low であるときの 2 種類を求める）．

以上により，NIC が同一でない場合であっても，消費電力の低い順にパスの割当てが可能となる．なお，パスセットを 2 種類求めるのは，Low か High かによってパスセットの並び，つまり消費電力の低い順が異なることが考えられるためであり，4.2.2 項 (4) (b) で述べたような全リンクが High 状態であるパスを複数用いるような状況では，High のパスセットからパスを取り出して割当てを行う．以上を拡張し，全リンクの状態を同一にするのではなく，一部のリンクの状態を High にして，複数のパスでこれを共有することも可能だが，リンクの状態の組合せ数はリンク数にともなって指数関数的に増加するため，最適な組合せを実用的な時間で得ることは困難である．

また，本論文では，既存の通信機器への適用の容易さを考えるにあたり，OpenFlow の最も古い安定版であり，広く商用化・導入もされている OpenFlow 1.0 を採用しているが，OpenFlow 1.0 は，コントローラがスイッチからの Packet In を契機に処理を開始する，リアクティブ型の OpenFlow である．そのため，本論文の実験では，MPTCP および UDP のトラフィックしか流していないが，実際のネットワークでは様々なトラフィックが流れるため，それらによって大量の Packet In が発生し，コントローラの処理を圧迫するおそれがある．この対策としては，OpenFlow のバージョンを 1.3 以上にし，リアクティブな Packet In の発生を限定する方法が考えられる．特に，OpenFlow 1.5 からは，TCP の制御フラグもフローエントリのマッチ条件に指定できる．現在はコネクションの切断・終了を Flow Removed メッセージで判断しているが，この方式では，実

際の切断・終了からタイムアウトによってフローエントリが削除されるまでにタイムラグが発生し，その間に活用されない帯域が発生する問題がある．TCP の制御フラグをマッチ条件に指定できれば，コネクションの切断・終了をより正確に検知できるようになり，活用されない帯域の発生を防ぐことができるようになる．

加えて，図 11 で示したとおり，フローエントリのインストールの際に通信中断が発生するときがあるため，この問題についても対策を講じる必要がある．ただし，本論文の実験は，PC で構成したスイッチを用いた環境で行っているため，OpenFlow 専用スイッチを用いた場合，この中断時間は改善される可能性がある．

最後に，本論文では，2.3.1 項で示した ALR の課題のうち，ノードが行う通信を考慮した帯域切替えについて解決できていない．この課題に対する対策としては，各コネクションに対して一律にパスを割り当てる方法から，各コネクションの DSCP などの情報から QoS を考慮してパスを割り当てる方法に変更することが考えられる．

謝辞 本論文の実験を行うためのリサイクル PC をご提供いただきました筑波大学システム情報系技術専門職員の雨谷恵様には，突然のお願いにもかかわらず，提供をご了解いただけましたことを深く感謝いたします．

参考文献

[1] 総務省：我が国のインターネットにおけるトラヒックの集計結果（2017 年 11 月分），総務省（オンライン），入手先 ⟨http://www.soumu.go.jp/main_content/000535404.pdf⟩（参照 2017-03-05）．

[2] 米津 遥，石井大介，岡本 聡ほか：自己組織化省エネルギーネットワーク MiDORi における消費電力最適化のためのトポロジー計算手法，電子情報通信学会論文誌 B，Vol.J94-B, No.10, pp.1323–1331 (2011).

[3] 平尾明子，竹下秀俊，岡本 聡ほか：省電力ネットワーク MiDORi における Point to Multi-point 対応ルーチング高速計算手法，電子情報通信学会論文誌 B，Vol.J97-B, No.2, pp.99–109 (2014).

[4] Christensen, K., Reviriego, P., Nordman, B., et al.: IEEE 802.3az: The Road to Energy Efficient Ethernet, *IEEE Communications Magazine*, Vol.48, No.11, pp.50–56 (2010).

[5] Reviriego, P., Maestro, J. and Larrabeiti, D.: Burst Transmission for Energy-efficient Ethernet, *IEEE Internet Computing*, Vol.14, No.4, pp.50–57 (2010).

[6] Gunaratne, C., Christensen, K. and Nordman, B.: Managing Energy Consumption Costs in Desktop PCs and LAN Switches with Proxying, Split TCP Connections, and Scaling of Link Speed, *International Journal of Network Management*, Vol.15, No.5, pp.297–310 (2005).

[7] Gunaratne, C., Cristensen, K., Nordman, B., et al.: Reducing the Energy Cunsumption of Ethernet with Adaptive Link Rate (ALR), *IEEE Trans. Comput.*, Vol.57, No.4, pp.448–461 (2008).

[8] Zhang, B., Sabhanatarajan, K., Gordon-Ross, A., et al.: Real-time Performance Analysis of Adaptive Link Rate, *Proc. 33rd IEEE Conf. Local Computer Networks*, pp.282–288 (2008).

[9] Nishiguchi, M. and Kimura, S.: Communication Link Switching Method Based on Destination IP Address for Power Savings, *Proc. 4th International Symposium on Computing and Networking*, pp.343–346 (2016).

[10] Gupta, M. and Singh, S.: Greening of the Internet, *Proc. ACM Conf. Applications, Technologies, Architectures, and Protocols for Computer Communications*, pp.19–26 (2003).

[11] Gupta, M. and Singh, S.: Using Low-power Modes for Energy Conservation in Ethernet LANs, *Proc. 26th Annual IEEE Conf. Computer Communications*, pp.2451–2455 (2007).

[12] Mostowfi, M. and Christensen, K.: Saving Energy in LAN Switches: New Methods of Packet Coalescing for Energy Efficient Ethernet, *Proc. 2011 International Green Computing Conference and Workshops*, pp.1–8 (2011).

[13] Zhang, M., Yi, C., Liu, B., et al.: GreenTE: Power-aware Traffic Engineering, *Proc. IEEE International Conf. Network Protocols*, pp.21–30 (2010).

[14] Anand, H., Reardon, C., Subramaniyan, R., et al.: Ethernet Adaptive Link Rate (ALR): Analysis of a MAC Handshake Protocol, *Proc. 31st IEEE Conf. Local Computer Networks*, pp.533–534 (2006).

[15] Gunaratne, C., Cristensen, K. and Suen, S.: Ethernet Adaptive Link Rate (ALR): Analysis of a Buffer Threshold Policy, *Proc. 2006 Global Telecommunications Conf.*, pp.1–6 (2006).

[16] Gunaratne, C. and Cristensen, K.: Ethernet Adaptive Link Rate (ALR): System Design and Performance Evaluation, *Proc. 31st IEEE Conf. Local Computer Networks*, pp.28–35 (2006).

[17] Reviriego, P., Huiszoon, B., L'opez, V., et al.: Improving Energy Efficiency in IEEE 802.3ba High-rate Ethernet Optical Links, *IEEE Journal of Selected Topics in Quantum Electronics*, Vol.17, No.2, pp.419–427 (2011).

[18] Fisher, W., Suchara, M. and Rexford, J.: Greening Backbone Networks: Reducing Energy Consumption by Shutting Off Cables in Bundled Links, *Proc. 1st ACM SIGCOMM Workshop Green Network*, pp.29–34 (2010).

[19] Ford, A., Raiciu, C., Handley, M., et al.: Architectural Guidelines for Multipath TCP Development, RFC 6182 (2011).

[20] Ford, A., Raiciu, C., Handley, M., et al.: TCP Extensions for Multipath Operation with Multiple Addresses, RFC 6824.

[21] Paasch, C., Duchene, F. and Detal, G.: MultiPath TCP - Linux Kernel Implementation, available from ⟨https://www.multipath-tcp.org/⟩.

[22] van der Pol, R., Boele, S., Dijkstra, F., et al.: Multipathing with MPTCP and OpenFlow, *Proc. 2012 SC Companion High Performance Computing, Networking, Storage and Analysis*, pp.1617–1624 (2012).

[23] Nakasan, C., Ichikawa, K., Iida, H., et al.: A Simple Multipath OpenFlow Controller Using Topology-based Algorithm for Multipath TCP, *Concurrency and Computation: Practice and Experience*, Vol.29, No.13 (2017).

[24] Bista, B., Takanohashi, M., Takata, T., et al.: A Power Saving Scheme for OpenFlow Network, *Journal of Clean Energy Technologies*, Vol.1, No.4 (2013).

[25] Markiewicz, A., Tran, P. and Timm-Giel, A.: Energy Consumption Optimization for Software Defined Networks Considering Dynamic Traffic, *Proc. IEEE 3rd International Conf. Cloud Networking*, pp.155–160 (2014).

[26] Rodrigues, B., Riekstin, A., Januario, G., et al.: GreenSDN: Bringing Energy Efficiency to an SDN Emulation Environment, *Proc. 2015 IFIP/IEEE International Symposium on Integrated Network Management*, pp.948–953 (2015).
[27] Open vSwitch: OVS – Open vSwitch, available from ⟨https://www.openvswitch.org/⟩.
[28] Trema: Trema – Full-stack OpenFlow Framework in Ruby, available from ⟨https://trema.github.io/trema/⟩.
[29] 美濃圭佑，木村成伴：異種混合ネットワークにおけるサブフロー間の RTT 比を考慮した MPTCP のためのパケットスケジューリング方式，信学技報，Vol.114, No.401, pp.1–6 (2015).

西口 雅人 （正会員）

1993 年生．2016 年筑波大学情報学群情報メディア創成学類卒業．2018 年同大学大学院博士前期課程修了．同年日本電信電話株式会社 NTT ネットワークサービスシステム研究所入社．キャリアネットワーク仮想化に向けた研究開発に従事．

木村 成伴 （正会員）

1967 年生．1990 年東北大学工学部情報科学科卒業．1992 年同大学大学院博士前期課程修了．1995 年同博士課程修了．博士（情報科学）．同年筑波大学講師．2001 年同大学助教授．2007 年同大准教授．プロセス代数，ネットワークプロトコル，通信システムの効率評価等に関する研究に従事．電子情報通信学会，ソフトウェア科学学会，IEEE，IEEE-CS，IEEE-ComSoc，ACM 各会員．