DPN環境における外部情報に基づくOpenFlowを用いたプログラマブルなネットワーク制御方式

柳田 晴香[1]　中尾 彰宏[2]　山本 周[2]　山口 実靖[3]　小口 正人[1]

概要：近年のデジタル情報量の爆発的な増加に伴い，ネットワークトラフィック量も急増している．このような現在の状況で大規模な自然災害などが発生すると，トラフィック処理基盤にはバースト的な負荷変動がかかる．このような状況下では，従来の人手による静的な制御では対応しきれず，一部でネットワークが繋がりにくくなるといった問題が生じる．緊急災害時にネットワークが機能しなくなる事は致命的である．そこで SDN(Software-Defined Networking) や OpenFlow プロトコルの利用により，ソフトウェアによる自動ネットワーク制御を考える．しかし，現状の SDN ではデータプレーンはプログラマブルではないため，詳細な制御を行うことは難しい．緊急災害時においては，情報交換を行うアプリケーションのトラフィックを優先的に扱うなどきめ細やかな制御が必要となる．そこで本研究では，SDN を一歩進めた DPN(Deeply Programmable Network) の概念を用いて，アプリケーション毎の QoS 等を実現するより高度なトラフィック制御を行う．またこのような制御を，実社会を濃く反映する Twitter 等の外部情報をもとに行うことで，迅速な対応を可能にし，緊急災害時にも耐えうる安定した情報処理基盤の構築を目指す．本稿では，DPN 環境で OpenFlow の機能とトラフィック弁別を行う機能の両方を実装可能な FLARE によるネットワーク環境を構築し，提案システムの実装を行う．

Traffic Control Using External Information in a Deeply Programmable Network

Haruka YANAGIDA[1]　Akihiro NAKAO[2]　Shu YAMAMOTO[2]　Saneyasu YAMAGUCHI[3]　Masato OGUCHI[1]

1. はじめに

近年，デジタル情報量の爆発的な増加や携帯型デバイスの発展に伴い，ネットワークトラフィック量が急増し，その量は今後 5 年で約 13 倍になるとも言われている [1]．このような現在の状況で大規模な自然災害などが発生すると，トラフィック処理基盤にはバースト的な負荷変動がかかる．従来のネットワーク制御は，トラフィックの急増が予想される場合に，回線やサーバを事前に増強したり，経路の冗長化を行うことで耐障害性を保っている [2]-[3]．しかし，これは予測が極めて困難な緊急災害時においては対応が難しい．また，今後トラフィック量が急増すると，冗長化したネットワーク機器の管理も膨大なものとなり，このような静的な制御では限界がくると予想できる．実際に 2011 年 3 月に発生した東日本大震災において，ネットワークが断絶し，社会的な災害対策の重要性が再認識される事態となった [4]．よって，人手を介さず柔軟に制御できる動的なシステムが必要となる．

そこで SDN(Software-Defined Networking) や OpenFlow プロトコルでの，ソフトウェア自動制御を考える．しかし，現状の SDN ではデータプレーンはプログラマブルではないため，詳細な制御を行うことは難しい．緊急災害時においては，情報交換を行うアプリケーションのトラフィックを優先的に扱うなどきめ細やかな制御を行いたい．さらに，近年のビデオトラフィックの著しい増加が，さらなるネットワークトラフィックの混雑をもたらすとして懸念されている [1]．よって，緊急時においてはエンターテイメント目的の動画視聴のアプリケーションのトラフィックには逆に制限をかけるといったことが必要だと考えられる．

[1] お茶の水女子大学
2-1-1 Otsuka, Bunkyou-ku, Tokyo, 112-8610 JAPAN
[2] 東京大学
7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-8654 JAPAN
[3] 工学院大学
1-24-2 Nishi-Shinjuku, Shinjuku-ku, Tokyo, 163-8677 JAPAN

SDN ではネットワーク機器をソフトウェアで自動的に制御するが，ネットワーク機器部分がプログラマブルではないため，ネットワーク機器自体に従来の制御以上の機能を持たせることは難しい．よってこのような詳細な制御は不可能である．

そこで本研究では，SDN を一歩進めた DPN(Deeply Programmable Network) の概念を用いて，アプリケーション毎の QoS 等を実現するより高度なトラフィック制御を行う．DPN ではデータプレーンをもプログラマブルにし，ネットワーク制御をフルにプログラム可能にする．また，文献 [5] の Twitter と外部情報から障害情報を早期検知するシステムと合わせて制御を行うことで，迅速な対応を可能にする．これらによって，イベント時におこるバースト的な負荷変動に柔軟かつ迅速に対応し，緊急災害時にも耐えうる安定した情報処理基盤の構築を目指す．本稿では，DPN 環境で OpenFlow の機能とトラフィック弁別を行う機能の両方を実装可能な FLARE [6] によるネットワーク環境を構築し，提案システムの実装を行う．

本論文の構成は以下の通りである．まず 2 章で関連研究について述べ，3 章で SDN と DPN の概念を紹介する．4 章で研究方針を説明し，5 章で提案システムの概要と実験について述べる．6 章では FLARE 実機実験の実装や検討内容を紹介し，7 章で本論文をまとめる．

2. 関連研究

SDN や OpenFlow 技術を利用したトラフィック制御は既に，クラウドデータセンタでは実用化の段階にある [7]．企業ネットワーク向け SDN もトライアルから実用期に入り，少なくとも 2017 年までにネットワーク市場の 3 割が SDN 化し，世界市場規模は 9 倍以上に拡大するとも言われている [8]．それに伴い SDN や OpenFlow 技術を利用した障害対策手法も提案されてきた [9]．また，制御のための障害検知に Twitter を利用する手法も提案されている [10]-[11]．

しかし，どれもソフトウェアによるネットワーク機器の動的管理という SDN の概念に留まっている．これからの実社会のダイナミックな状況変化に適切に対応するには，この現状の SDN では実現が難しい詳細なレベルの制御が必要となることが考えられる．例えば緊急災害時における，アプリケーション毎の QoS の実現である．既存の SDN の枠組みでは，通話やメールといった重要通信とそれ以外とを識別する機能がない．

よって，SDN を拡張し，データプレーンのプログラム可能性を追求する動きが出てきた．コントロールプレーンより上のプログラム可能性だけでは限定的な制御しか出来ないことが認知されつつある [12]．東京大学中尾研究室では，この DPN の概念によりデータプレーンのプログラム可能性を実現した FLARE プロトコルの研究が行われている [6]．この FLARE では OpenFlow の機能とトラフィック弁別を行う機能の両方を実装可能である [13]．

そこで，本研究では，制御のための障害検知に Twitter を利用した，FLARE によるトラフィック制御方式を提案し，その後半部分の実装を行う．

3. SDN と DPN

3.1 既存の SDN/OpenFlow による制御

OpenFlow の仕組みは，コントロールプレーンとデータプレーンの機能の分離である．コントロールプレーンとはデータの転送経路を決定する経路制御の頭脳であり，データプレーンはコントロールプレーンの指示に従ってデータを転送する．これら二つの機能は，従来の物理スイッチでは図 1 のようにどちらも同じ物理機器内に組み込まれていたが，OpenFlow ではこれらを図 2 のように分離した．コントロールプレーンをネットワーク機器外部のサーバ上にソフトウェアとして分離し，スイッチではデータ転送機能のみを実行する仕組みである．OpenFlow プロトコルはこれらを接続する標準的なインタフェースで，インタフェース経由でのデータプレーンの制御を可能にする．これにより，ネットワーク全体を俯瞰したトラフィックの制御を，ソフトウェアコントローラでプログラマブルに行うことが可能になる．

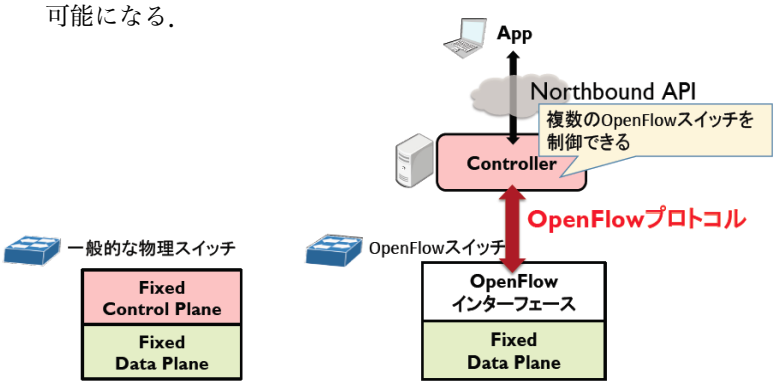

図 1 従来のスイッチ　　**図 2** OpenFlow の仕組み

3.2 DPN/FLARE による制御

3.1 節より，SDN ではデータプレーンを持つスイッチはデータ転送という決められた動きしか実行しないことがわかる．このデータプレーン部分をもプログラム可能にするのが，DPN の概念である．そして DPN を実現するために開発された技術の一つが，中尾研究室で研究中の FLARE アーキテクチャである [6]．この技術によって，データプレーンを持つスイッチ部分に，必要な機能を自由に付加することが可能になる．従って，従来のネットワーク機器ではなかった機能を持つような，新たなネットワーク機器を自由に作成可能になる．

FLARE では，Click というプログラミング言語でデータプレーンを記述する．Click はネットワーク機器をソフトウェアで再現する言語で，米国の大学ではネットワークの研究や教育などで利用されている．Click の特徴としては，

「フレームを受け取る」「フレームを転送する」「ルーティングテーブルを参照する」といった様々な基本機能が，モジュールとして用意されている点である．モジュールを組み合わせることで，簡単にスクリプトでネットワーク機器の動作を記述できる．また，独自のモジュールを作成することも可能で，ユニークな動作をするネットワーク機器を作成することが可能である [14].

東京大学では，FLARE 向けに OpenFlow に対応した Click のモジュールを独自に開発しており，FLARE の OpenFlow による操作を可能にしている．FLARE ノードは，物理ノード上にスリバーと呼ばれるコントロールプレーンとデータプレーンを搭載した仮想スイッチを提供することができる．それら仮想スイッチの組合わせでスライスと呼ばれる，仮想ネットワークを作成できる (図 3).

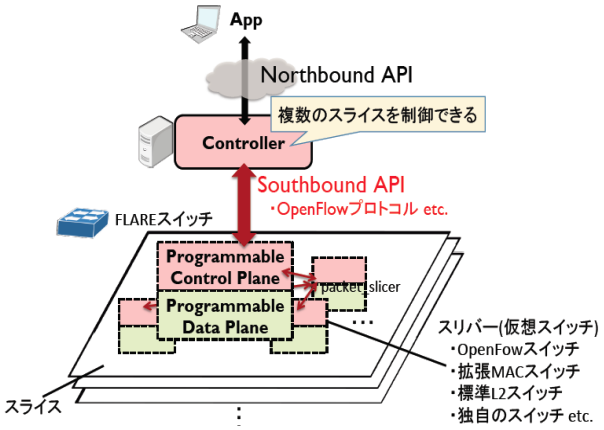

図 3　FLARE の仕組み

3.3　SDN と DPN の比較

ここで SDN と DPN についてまとめると，図 4 の比較より FLARE のほうがよりディープにネットワークをプログラム可能にしているということがわかる．

	OpenFlow	FLARE
コントロールプレーンプログラム性（経路制御・ネットワーク管理）	○	○
データプレーンプログラム性（パケット処理）	×	○
アクセス可能な層	トランスポート層より下位層（L1〜L4）	アプリケーション層含む全ての層（L1〜L7）

図 4　SDN と DPN の比較

OpenFlow スイッチを含む一般的なスイッチでは，フレームやパケットのヘッダの一部分 (L1〜L4) だけを見て転送処理を行う．それに対し FLARE スイッチはデータ部を含むどの部分 (L1〜L7) でも読み取ったり変更を加えたりできる (図 5).

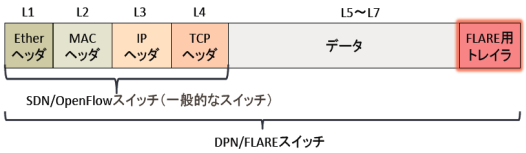

図 5　SDN と DPN スイッチのアクセス範囲

よって，ホスト側のカーネルに事前に変更を加え，パケットに例えばアプリケーションの情報など制御に使いたい情報をトレイラとして付けるようにすると，このパケットが FLARE スイッチへ入ってくれば，FLARE はトレイラにアクセスすることが可能なので，どのようなアプリケーションを使用しているのか検出が可能になる．これにより，従来では出来なかった，アプリケーションの種別に基づく制御が実現可能になる．

4.　研究方針

前章で説明してきた DPN 環境でのアプリケーション毎の QoS を実現する前に，①まず，障害時における自動経路切替制御を SDN 環境で OpenFlow を用いて行う．②次に，同じ制御を FLARE 実機上の DPN 環境に実装する．③そして，FLARE 実機上でアプリケーションの振り分けを行うプロトコルを実装し，④最後に障害時におけるアプリケーション毎の QoS を実現する．以上のステップを踏むことで，本研究の目的達成を目指す．

5.　提案システム

5.1　提案システム概要

4 章の①を実現するための提案システムの概要を図 6 に示す．この提案システムにより，障害時における自動経路切替制御を実現する．動作は以下の通りである．

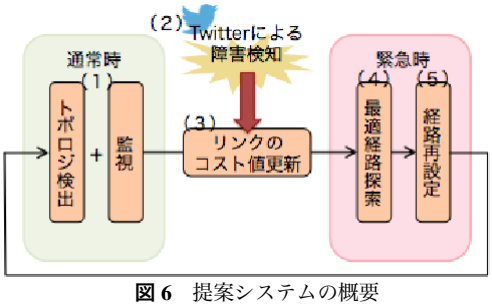

図 6　提案システムの概要

(1)　トポロジ検出と監視
スイッチのポートの接続状況を検出するアプリケーションを動かし，トポロジ情報の検出を行う．また，通常時において，トラフィックモニタによりネットワークに異常がないかの監視を行う．

(2)　Twitter による障害検知
1 章で説明したように，これは文献 [5] により検討されている．本研究ではこの文献より抽出された障害の位置情報等を用いてどの辺りでどのくらいの規模の障害

がおこっているのかを取得し，(1) で行うトラフィックモニタの情報と照らし合わせ，障害の場所を特定する．

(3) リンクのコスト値更新
(2) で特定した障害情報に従い，スイッチ間を結ぶリンクのコスト値を更新する．

(4) 最適経路探索
更新されたコスト値を用いて，ダイクストラ法で最適経路を探索する．

(5) 経路再設定
決定された最適経路に経路を再設定する．

また，これら一連の動作は自動的に行うものとする．次節の後半で具体的な動作の説明を行う．

5.2 Mininet による提案システムの実装

提案システムの実装を，まず Mininet 仮想環境で行う．Mininet は OpenFlow フレームワークの一つである，ネットワークエミュレータで，実際に物理スイッチやサーバを用意しなくても擬似的に OpenFlow ネットワークを構築できる．Mininet で開発したスクリプトは FLARE と互換性があるため，こちらを利用する．PC に VirtualBox をインストールし，仮想環境で OpenFlow による制御モデルを構築し開発を行った．この開発環境を表 1 に示す．

表 1 mininet による開発環境

OS	ubuntu14.04 64bit
フレームワーク	Mininet 2.1.0p1
コントローラ	Ryu-manager 3.15
スイッチ	Open vSwitch 2.0.2

上記の環境で，以下の開発を行い，提案システムの実装を行った．

(1) トポロジ作成と通信経路設定 (図 7)
各スイッチにそれぞれひとつずつホスト h1〜h4 が接続された，4OVS(Open vSwitch) のメッシュトポロジをスクリプトより作成した．図 8 以降ではホストを省略している．通信経路は，REST-API を用いてフローテーブルを作成し，h1 から h2 へ h1-s1-s2-h2 で設定した．

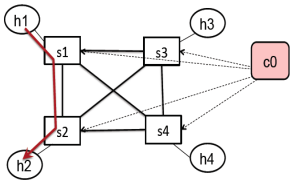

図 7 4OVS のメッシュトポロジ

(2) トポロジ検出 (図 8)
通常時においてトポロジを検出できるよう，LLDP(Link Layer Discovery Protocol) パケットを使ってトポロジ検出アプリケーションを作成した．

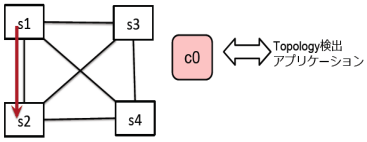

図 8 トポロジ検出アプリケーション

(3) トポロジ情報にコスト値を付加 (図 9)
(2) で検出したトポロジ情報に，コスト値を付加するスクリプトを作成した．コスト値はデフォルトで 1 となるよう設定した．

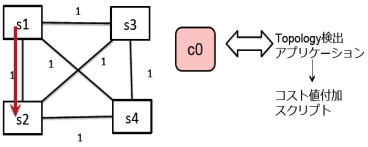

図 9 コスト値付加スクリプト

(4) コスト値の更新 (図 10)
輻輳が発生したという情報が入ってきたと仮定し，手動でコスト値の更新を行う．

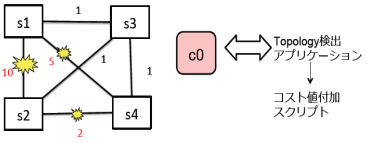

図 10 輻輳が発生した場合

(5) 最適経路探索 (図 11)
コスト値ををもとに，ダイクストラ法による最適経路探索を行うアプリケーションを作成した．ここで，コスト値が小さいということは帯域が大きいということを示す．

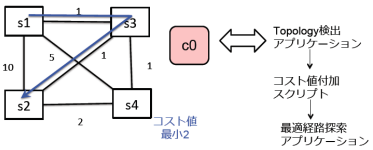

図 11 ダイクストラによる最適経路探索アプリケーション

(6) 経路の再設定 (図 12)
最適経路に通信経路を設定しなおすスクリプトを，REST-API を用いて作成した．

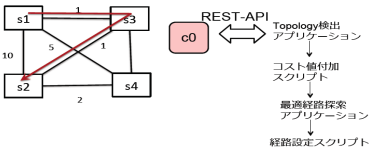

図 12 経路設定スクリプト

(7) アプリの連携によるシステムの自動化 (図 13)
一連の流れを連携させ，障害のインプットを与えると自動的に最適経路に切替えるよう，システムの自動化を図る．

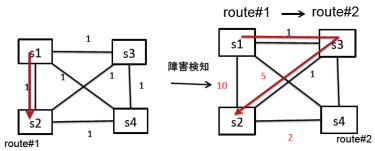

図 13　障害イベントによる自動経路切替え

以上により実装された提案システムの具体的な動作を説明する．通常時にトポロジ検出をおこなうことでコントローラはスイッチのトポロジ情報を保持している．ここで，文献 [5] の Twitter による障害検知システムより，s1-s2, s1-s3, s2-s3 の経路で障害を検知したと仮定すると，それぞれのリンクのコスト値をトラフィック量に基づき更新する．そして，ダイクストラ法が実行され，コスト値最小の経路を探索するため，最適経路はコスト値 2 である s1-s4-s2 の経路と決定される．よって，元の s1-s2 の経路から s1-s4-s2 の経路に再設定されることで，帯域の大きい最適な経路に自動的に切替が行われる．

5.3　疑似輻輳による提案システムの動作確認

Mininet で構築したシステムの動作の検証をするために，輻輳情報を手動で入力し経路を再設定させる実験を行った．5.2 節 (4) の様に s1-s2 間に 輻輳が発生したとして仮定し，輻輳情報を実装システムに手動で入力した．結果，提案システムの動作を確認することができた (図 14)．

図 14　障害イベントによる自動経路切替えの確認

図 14 中の青い線は輻輳の前後を表している．この線の前後でスイッチのポート番号が切り替わったことで，経路が route#1 から route#2 へ切り替わったことが確認できる．

6.　FLARE 実機実験

本研究では，図 15 に示す物理構成で実機実験を行う．FLARE Central は FLARE 管理用のサーバである．このサーバ上でスリバーやスライスの作成等，FLARE スイッチの管理を行う．本研究ではこの FLARE Central サーバ上にコントローラを置く．

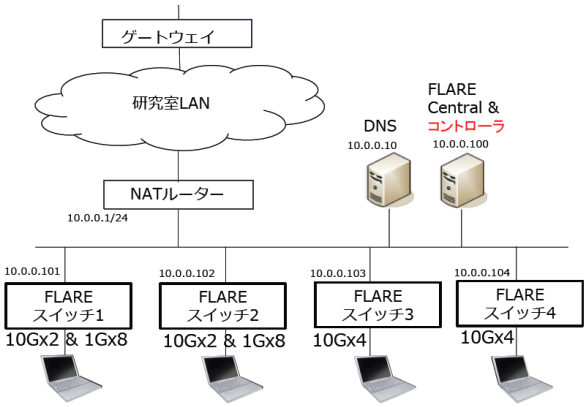

図 15　FLARE 物理構成図

6.1　FLARE による提案システムの実装

4 章の②を実現するため，5.1 節の提案システムを FLARE 実機上の DPN 環境に実装する．以下の手順で，提案システムの実装を行う．

(1) packet_slicer と flare_switch1 という名前のスリバーを作成する．VM の種類は flare-switch である．

(2) 各 FLARE ノード 1〜4 全てに作成したスリバーをアドする．これにより，スライスが作成される．

(3) 各スリバーのデータプレーンに click をインストールし，click プログラムを書き換えて，OpenFlow 対応のスリバーにする．

(4) (3) で作成した click プログラムをスリバーで動作させ，5.2 節で作成したプログラムをコントローラで動作させることで，FLARE 上で OpenFlow による自動経路切替制御を実現する．動作イメージは図 16 の通りである．

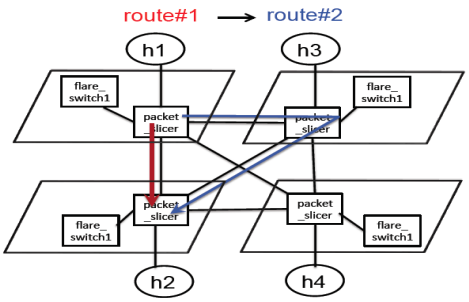

図 16　FLARE での障害イベントによる自動経路切替え

6.2　アプリケーション毎の QoS の検討

4 章の③の実現について検討を行う．6.1 節の (3) で作成する click プログラムに，トレイラを取り出しアプリケーションごとに分類するモジュールを付加させることで，アプリケーションの振り分けは実現できると考えられる．動作イメージは図 17 の通りである．

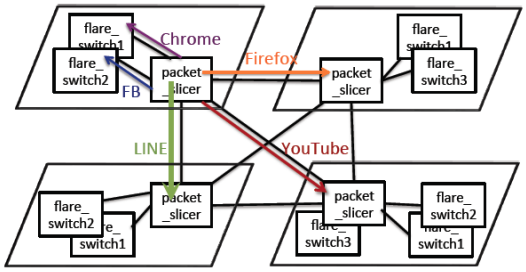

図 17 FLARE でのアプリケーションの振り分け

7. まとめと今後の課題

緊急災害時に Twitter 等の外部情報より早期検知された障害をトリガとして，ネットワークトラフィックの最適化をアプリケーション毎に自動で行う，高度なネットワークトラフィック制御システムを提案した．

現時点では，Mininet 仮想環境と FLARE 実機環境で，OpenFlow を用いたコンテキストに基づく経路切替制御を自動で行う，提案システムの挙動が確認できた．今後 FLARE での開発を進め，障害時におけるアプリケーション毎の QoS を実現するシステムを実装していく．また合わせて，性能評価を行っていく．

今後の課題としては，大きく 3 つある．1 つ目は通信設定を増やした時に，提案システムでは対応しきれないため，拡張した提案システムを模索しなければならない点である．2 つ目は大規模震災時における耐障害性として，コントローラが落ちてしまうことを想定していないことである．この点については，今後分散コントローラの検討を行う必要がある．3 つ目は外部情報とコントローラをつなげる部分の設計である．今後どのように設計していくか検討したい．

謝辞

本研究は一部，総務省戦略的情報通信研究開発推進事業 (SCOPE) 先進的通信アプリケーション開発推進型研究開発によるものである．

参考文献

[1] Cisco Visual Networking Index(VNI)：全世界のモバイルデータトラフィックの予測アップデート (ホワイトペーパ), http://www.cisco.com/web/JP/solution/isp/ipngn/literature/pdf/white_paper_c11-520862.pdf，2015 年 2 月 4 日

[2] NTT DOCOMO：どうなる！災害時の携帯電話, https://www.nttdocomo.co.jp/binary/pdf/info/news_release/report/050819.pdf，2005 年 8 月 19 日

[3] au by KDDI：災害対策への取り組み災害時・緊急時対策, http://www.au.kddi.com/mobile/anti-disaster/action/index01.html，2014 年 10 月

[4] NTT DOCOMO：https://www.nttdocomo.co.jp/binary/pdf/corporate/technology/rd/technical_journal/bn/vol20_4/vol20_4_026jp.pdf，テクニカル・ジャーナル Vol. 20 No. 4，2013 年

[5] 丸千尋，榎美紀，中尾彰宏，山本周，山口実靖，小口正人：大規模災害時における Twitter を用いたネットワークシステム制御に有用な情報の抽出，DEIM2015，C7-3, 2015 年 3 月

[6] 東京大学大学院情報学環中尾研究室：https://www.nakao-lab.org/nikkeibp-nakaolab.pdf

[7] 飯島明夫：OpenFlow/SDN のキャリアネットワークへの適用について，電子情報通信学会技術研究報告，NS，ネットワークシステム 112.231 (2012): 85-87.

[8] NEC：エンタープライズ向け SDN ソリューション事業, 2014 年 8 月 27 日

[9] NEC：OpenFlow による災害時でもつながるモバイル通信網の制御を実現, http://jpn.nec.com/rd/innovation/feature/2013/11-openflow.html?，コンセンサス，2013 年 11-12 月号

[10] 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，DEIM2014，C9-6，2014 年 3 月

[11] 高橋裕，秋山友理愛，神津智樹，山口実靖：バースト的負荷変動を考慮した OpenFlow を用いた動的資源割り当て (SDN/OpenFlow)，電子情報通信学会技術研究報告. NS, ネットワークシステム 113.472 (2014): 225-229.

[12] Akihiro Nakao：Software-Defined Data Plane Enhancing SDN and NFV，Special Section on Quality of Diversifying Communication Networks and Services, IEICE Transactions on Communications , Vol.E98-B, No.1, pp12-19, January 2015

[13] Akihiro Nakao, Ping Du：Application and Device Specific Slicing for MVNO，2014 International Science and Technology Conference (Modern Networking Technologies) (MoNeTeC), 2014, October

[14] The Click Modular Router Project：http://www.read.cs.ucla.edu/click/

[15] Mininet An Instant Virtual Network on your Laptop (or other PC)，http://mininet.org/