DPN環境におけるコンテキストに基づいたネットワーク制御方式

柳田　晴香† 　中尾　彰宏†† 　山本　周†† 　山口　実靖††† 　小口　正人†

† お茶の水女子大学
〒 112-8610 東京都文京区大塚 2-1-1
†† 東京大学
〒 113-8654 東京都文京区本郷 7-3-1
††† 工学院大学
〒 163-8677 東京都新宿区西新宿 1-24-2
E-mail: †{g1120541,oguchi}@is.ocha.ac.jp, ††{nakao,shu}@iii.u-tokyo.ac.jp, †††sane@cc.kogakuin.ac.jp

あらまし　震災のような緊急時には，被災地域で通常の何倍にも及ぶ通信が集中することで輻輳が発生し，ネットワークの機能が著しく低下してしまうといった問題が生じる．緊急災害時には，ネットワークによる情報提供が最も重要であるにも関わらず，これが機能しなくなる事は致命的である．現状のネットワーク制御では，経路を二重化し，一方の経路で障害が発生したら，もう一方の経路に切り替えるといった単純な経路制御や，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するといった制御しか行われていない．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．さらに，膨大で多種多様なデータが飛び交うようになった現代のネットワークでは，人が状況を目で見て判断する静的な制御では限界となってきた．そこで，本研究では細かい条件に応じた動的な制御を迅速に行うため，SDN(Software-Defined Networking) や OpenFlow プロトコルを用いて，ソフトウェアによる自動ネットワーク制御を行う．また，アプリケーションごとに緊急度の優先順位をつけるなどといった，より細かい制御を行うために，DPN(Deeply Programmable Network) の概念も適用するという点が従来の研究と異なる．本研究は外部情報より障害を検知したと仮定し，それをトリガとして，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手法の提案を行う．これにより，緊急災害時等でもユーザが安定して情報へアクセスできるシステムの構築を目指す．

キーワード　SDN，OpenFlow，DPN

Traffic Control Using Context in a Deeply Programmable Network

Haruka YANAGIDA†, Akihiro NAKAO††, Shu YAMAMOTO††, Saneyasu YAMAGUCHI†††, and Masato OGUCHI†

† Ochanomizu University
2-1-1 Otsuka, Bunkyou-ku, Tokyo, 112-8610 JAPAN
†† Tokyo University
7-3-1 Hongo, Bunkyo-ku, Tokyo, 113-8654 JAPAN
††† Kogakuin University
1-24-2 Nishi-Shinjuku, Shinjuku-ku, Tokyo, 163-8677 JAPAN
E-mail: †{g1120541,oguchi}@is.ocha.ac.jp, ††{nakao,shu}@iii.u-tokyo.ac.jp, †††sane@cc.kogakuin.ac.jp

1. はじめに

スマートフォンのような高機能なモバイル端末の発達とクラウドコンピューティングの普及などにより，ネットワーク上にビッグデータが蓄積されるようになってきた．これにはユーザによる情報発信，センサやカメラなどのデバイスによる出力など多様なデータが含まれる．その容量は増加し続けており，これらは基本的にインターネットを介してクラウドやデータセン

タに送り込まれる．またビッグデータからはさまざまな情報が生み出され，それを引き出すためのユーザからのアクセスもインターネット上を飛び交っている．すなわち現在のネットワークは，膨大で多種多様なトラフィックにより埋められ混沌とした状態になっている．

このようなネットワークを流れるトラフィックは一定ではなく，社会現象により大きく変動する．下記図 1 に示すように，何か大きな出来事やイベントなどがあれば，それにより発生するデータ量も増え，またこれに関心を持ってアクセスされるトラフィックも急増する．その最たるものが震災などの大規模災害時のトラフィックで，発生するビッグデータの量は桁違いに多くなり，情報を得ようとするユーザからのアクセス量も激増する．すなわちサイバー空間とフィジカルな世界（実社会）が密接に結び付くようになった現代では，ネットワーク上のトラフィックは，実社会の状況を色濃く反映したものになっている．そしてユーザは，実社会の変動時にこそネットワークから得られる情報を必要とする．

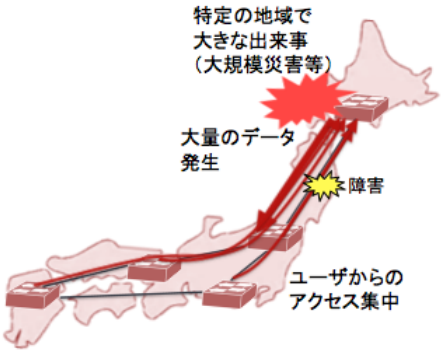

図 1　災害発生時の現在のネットワーク

従ってネットワークのトラフィック制御を考える場合には，社会状況を考慮して行うのが妥当である．[1]- [2] 従来においても，例えば大きなイベントが予定されトラフィックの急増が予想される場合には，これに対処可能なように回線やサーバを増強する事が行われてきた．しかし，そのように予想できる範囲であれば良いが，実社会の動きに基づくトラフィックの変動は，時に予想を大きく上回る可能性もある．さらに震災のように予測が極めて難しい大規模自然災害が起こったりする可能性もある．そのような場合には，トラフィックの変動が想定していた範囲を超えてしまい，ネットワークの機能が著しく低下してしまう．緊急災害時には，ネットワークによる情報提供が最も重要であるにも関わらず，これが機能しなくなる事は致命的である．

一方で，クラウドやデータセンタで扱われるデータの量や種類が増え，ネットワーク上の流れも複雑になるにつれ，これを人間が目で見て判断し人手で制御を行うのは限界となってきた．何らかの形で，制御の自動化が必要である．しかしながら現状のインターネットでは，単純な経路制御や回線容量の増減くらいしか制御の方法が無い．例えばどのようなアプリケーションによりどのようなトラフィックが発生し，これをどう制御したら良いかわかったとしても，現状の単純な制御機能のみでこれを実現するのは極めて難しい．さらに，緊急災害時などには特定のアプリケーションのトラフィックを優先的に扱うような対応が望ましいが，現状のネットワークでそのような制御をきちんと行う事はほぼ不可能である．

以上のように，現状のネットワークでは，実社会の変動に対応し，その際にユーザが必要とする情報を安定して供給する事が難しい．そこで本研究は，社会状況が大きく変化する出来事やイベントなどがあった場合に，その情報を感知し状況を把握してネットワーク制御を行い，それによって，実社会の変動時にもユーザが必要とする情報に安定してアクセスできるようにする仕組みを実現することを目的とする．

これらの実現を目指す事柄は，以下の 2 点に集約される．

i　実社会の状況，特にネットワークのトラフィックに大きな影響を与えそうなイベントを的確に捉え，どのような制御を行えば良いか判断する情報を抽出する

ii　実社会の状況から抽出された複雑な情報に基づき，これに対処する高度なトラフィック制御を実現する

i は文献 [3] により検討されている．本研究では，ii について特定のアプリケーションのトラフィックを優先的に扱う制御を設計する．さらに，事前に複数の仮想ネットワークをスライスとして構築しておき，特定の条件が満たされた場合に，スライスの切換えを行う事によって，ネットワーク全体を素早く緊急災害時のモードへ移行させたり等も可能となる．実現するネットワーク環境を以下の図 2 に示す．

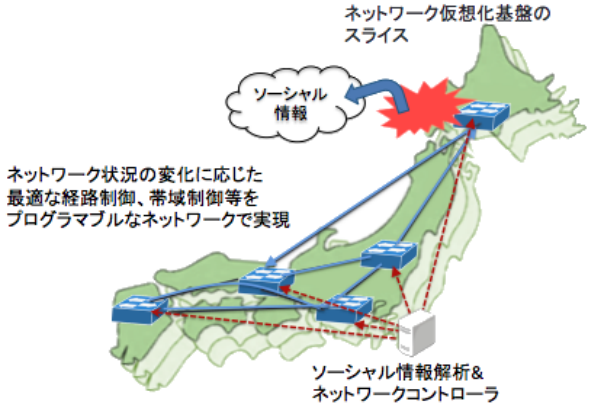

図 2　DPN 環境により実現するネットワーク

2. 関 連 研 究

これまでにも SDN や OpenFlow 技術を利用したソフトウェアによる自動トラフィック制御は実現されてきた [4]- [5]．また，Twitter 等の実社会の状況を濃く反映したモニタリング情報をトリガとした手法も提案されている [6]- [7].

しかし，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，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．すなわち緊急災害時において，現場の状況を伝えたり情報交換を行ったりするためのアプリケーションについては優先的にパケットを通し，エンターテインメント目的のトラフィックには制約をかけるなどといった，アプリケーションの種類に基づく制御である．OpenFlow など既存の SDN の枠組みでは，そこまで詳細な制御を実現する事は難しい．

そこで，本研究では，このような高度なトラフィック制御を実現するため，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する [8].

3. プログラマブルなネットワーク制御手法

3.1 SDN/OpenFlow による制御

SDN(Software-Defined Network) とはネットワークの構成，機能，性能などをソフトウェアの操作だけで動的に設定，変更できるネットワーク，あるいはそのためのコンセプトを指す．今まで遅れていた，ネットワークの仮想化を進めていこうという動きである．

そして，その SDN を実現するための要素技術の一つが OpenFlow である．OpenFlow では図 3 のように，ネットワーク全体の経路制御をコントローラと呼ばれる別の機器上のソフトウェアで集中管理し，スイッチではデータ転送機能のみを実行する．プログラムで自由に記述できるコントローラにより，ネットワーク全体を俯瞰したトラフィックの制御を自動的に行うことが可能になる．

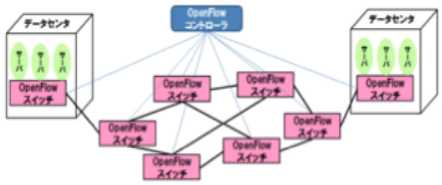

図 3　OpenFlow コントローラによるスイッチの一元制御

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 4 の左図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 ではこれらを右図のように分離した．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．

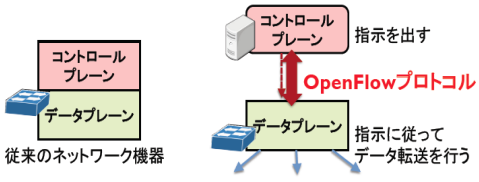

図 4　OpenFlow の仕組み

3.2 DPN/FLARE による制御

しかし，本研究が目指すところは，アプリケーションの種類に基づくきめ細やかな経路・帯域制御である．OpenFlow など既存の SDN の枠組みでは，このような詳細なレベルの制御を行うことは難しい．

そこで，DPN の概念と，東京大学中尾研究室で研究中の FLARE スイッチ [8] を導入する．DPN (Deeply Programmable Network) とは，よりディープにネットワークをフルプログラム可能にしようという，SDN をもう一歩進めた概念を示す．

そして，その 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．FLARE アーキテクチャは，コントロールプレーンに加えデータプレーンをもプログラム可能にする．この技術によって，SDN/OpenFlow で可能な制御に加え，アプリケーションの種類に基づく制御や，仮想化ネットワークの切替制御など，さらに詳細な制御が可能になる．

図 5 に SDN/OpenFlow と DPN/FLARE をまとめる．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）	×（スイッチはまだハードウェア）	○（スイッチもソフトウェア化）
アクセス可能な層	トランスポート層より下位層	アプリケーション層含む全ての層

図 5　SDN と DPN の比較

OpenFlow スイッチはハードウェアであり，トランスポート層より下位層のヘッダ情報を照合しつつデータ転送を行う．これに対し FLARE ノードは，物理スイッチ部分でさえも，プログラム可能なソフトウェアに近いハードウェア化することにより，アプリケーション層より上位層のデータにまでアクセス出来る．これにより，アプリの検出が可能になり ([9]- [10])，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簡単に実現可能になる．

また．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．(スイッチ上に並列処理可能な仮想マシン OS がいくつも存在する様な状態.) これにより，仮想化ネットワークの切替制御なども可能になる．

4. 提案システムの概要

4.1 提案システムの動作

本研究では，外部情報より障害を検知した際に，それをトリガとして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手法の提案を行う．提案システムの概要を図 6 に示す．動作は以下の通りである．

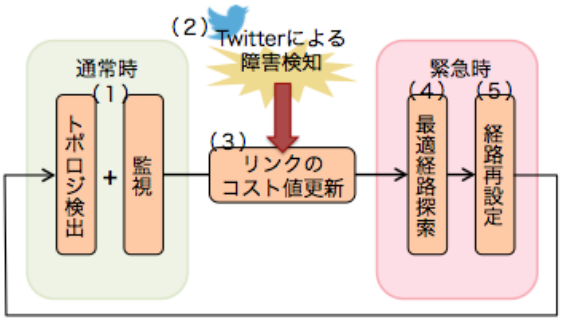

図 6　提案システムの概要

(1)　トポロジ検出と監視

どのスイッチのどのポートが他のスイッチのどのポートに繋がっているのかについて情報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について監視を行う．

(2)　Twitter による障害検知

1. 章の i で説明したように，これは文献 [3] 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情報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に従いどのノードでどのくらいのトラフィック量の障害が発生しているのか特定する．

(3)　リンクのコスト値更新

(2) で検知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

(3) で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

(4) で決定された最適経路に経路を再設定する．

さらに，緊急時においてネットワーク制御を人手を介さず迅速に行うという目的を達成するために，これら一連の動作の自動化を行う．

4.2　提案システムの検討

以上の提案システムの動作を以下の SDN と DPN の環境のもと，具体的な例を交えつつ，検討する．

①SDN/OpenFlow 環境

②DPN/FLARE 環境

まず，①の SDN/OpenFlow 環境では，図 7 や図 8 等が想定される．各スイッチに一つずつホストが接続されているが，この図では省略している．また，ホスト 1 からホスト 2 へとパケットが送信されているとする．まず，図 7 の単純な例で説明する．

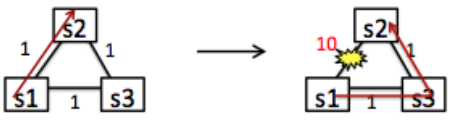

図 7　SDN 環境におけるスイッチ 3 つのメッシュトポロジの場合

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．この場合スイッチ 3 つが各ポートで各スイッチに繋がっており，それぞれのリンクのコスト値はデフォルトで 1 と設定される．ここで，s1-s2 の経路で障害を検知したと仮定すると，s1-s2 間のリンクのコスト値をトラフィック量に基づき 10 に更新する．そして，ダイクストラ法を実行すると，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-s3-s2 の経路と決定される．よって，元の s1-s2 の経路から決定された s1-s3-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．

次に，図 7 よりもやや複雑なトポロジである図 8 の例で検討する．この場合，障害検出で s1-s2，s1-s3，s1-s6，s2-s3，s2-s6 の 5 つの経路で輻輳等の障害が検知されたと仮定している．この際も，提案システムのダイクストラ法によって，コスト値 4 の最適経路が選択される．

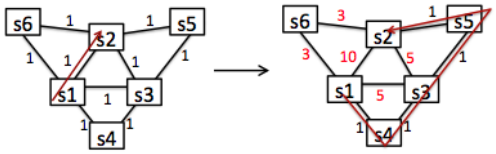

図 8　SDN 環境におけるスイッチ 6 つのメッシュトポロジの場合

②の FLARE スイッチを用いた DPN 環境では①で可能な制御に加え，図 9 や図 10 のような制御も可能になる．

FLARE スイッチを用いた DPN 環境では，仮想スイッチの組み合わせでスライスと呼ばれる仮想ネットワークを作成することができる．各スライスに対して，提案システムを適応させていく．これによって，図 9 のように，複数の仮想ネットワークをスライスとして構築しておき，特定の条件が満たされた場合に，スライスの切換えを行う事によって最適な経路を選択することができる．これが想定される具体的な例としては，障害を検知した際に，事前に作成された仮想ネットワークの切替によって，ネットワーク全体を素早く緊急災害時のモードへ移行する場合などがあげられる．

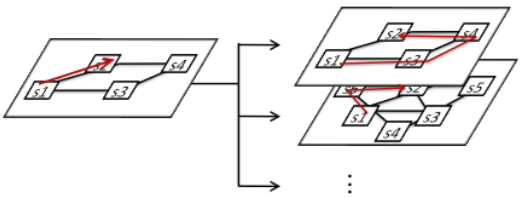

図 9　DPN 環境におけるスライスの切替

さらに，図 10 のように，アプリケーション毎の QoS を実現することができる．これが想定される具体的な例としては，緊急災害時において，現場の状況を伝えたり情報交換を行ったりするためのアプリケーションについては優先的にパケットを通し，エンターテインメント目的のトラフィックには制約をかけるなどといった制御である．

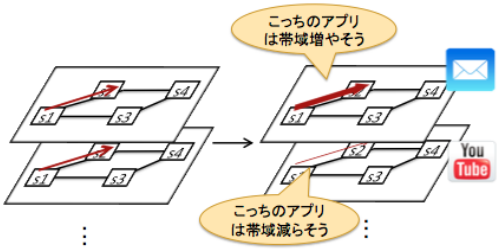

図 10　DPN 環境におけるアプリケーション毎の QoS の実現

5.　研 究 方 針

研究方針としては，以下の通りである．

①PC 上の Mininet 仮想環境で SDN/OpenFlow 開発

②ローカルな環境に FLARE スイッチを実装した
　実機環境を構築

③FLARE 実機環境で①で開発したスクリプトによる
　SDN/OpenFlow 開発

④FLARE 実機環境で DPN 開発

⑤広域なテストベット上で①～④を検証

次章より，上記の流れで行った研究の報告をする．

6.　Mininet を用いた制御プログラムの開発

本研究では，OpenFlow フレームワークの一つである Mininet [11] を用いて実機による実験の前に小実験を行った．この Mininet というフレームワークは機能として，ネットワークエミュレータツールを持っている．通常 OpenFlow のテストをする場合，実際に物理的な OpenFlow スイッチやパケットを送信するホスト等が必要となってくる．しかし Mininet は，ネットワークの定義ファイルを作ることで，仮想 OpenFlow スイッチや仮想ホストも設定できるため，OpenFlow コントローラの作成だけではなく，仮想環境で OpenFlow の動作を確認することができる．これを利用し，PC 上の仮想環境で OpenFlow による制御モデルを構築し実験を行った．この実験の環境を表 1 に示す．

表 1　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

6.1　Mininet における実験

上記の環境で，図 11 のような 3OVS ノードのメッシュトポロジを作成し，以下の 6 つの動作を確認した．

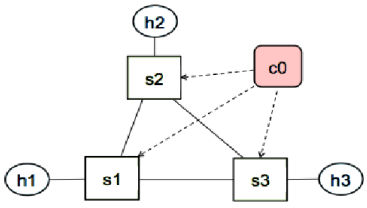

図 11　3OVS ノードのメッシュトポロジ

尚，h1 から h2 へ h1-s1-s2-h2 の経路で通信を設定している．

(1)　トポロジ検出

コントローラが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．

(2)　トポロジ情報にコスト値を付加

(1) 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．

(3)　コスト値の更新

h1-s1-s2-h2 の経路に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．

(4)　最適経路探索

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．

(5)　経路の再設定

ダイクストラ法によって求められた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．

(6)　アプリの連携によるシステムの自動化

以上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．

動作イメージは図 12 の通りである．

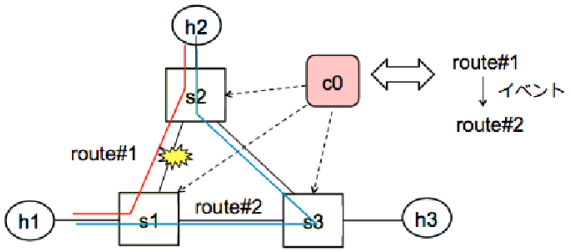

図 12　障害イベントによる自動経路切替え

7.　FLARE 実機実験

次に，ローカルな環境に FLARE スイッチを実装した実機環境を構築した．本研究では，図 13 のような物理構成で実機実験を行う．

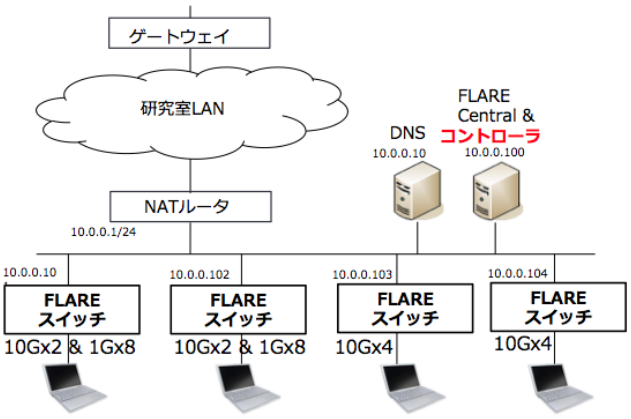

図 13　FLARE 実験環境

4 台の FLARE スイッチ各々に，端末が接続される形である．図中の FLARE Central は FLARE 管理用のサーバで，ここでス

ライスの作成等ができる．この FLARE Central サーバ上にコントローラを置く．コントローラで 4 台の FLARE スイッチを制御し，様々なトラフィック制御モデルを検討する．FLARE スイッチ 1 と 2 にはそれぞれ，1G のポートが 8 個と 10G のポートが付いていて，FLARE スイッチ 3 と 4 には 10G のポートがそれぞれ 4 個ずつ付いている．これらのポートをつないで，メッシュトポロジを作成することもできる．

まず動作させるプロトコルとして，6. 1 節に示した Mininet で開発した OpenFlow のスクリプトをそのまま乗せる形で動かす．次に，アプリケーション毎のトラフィックの制御等 DPN を生かした制御を行っていく．そして最終的には，それらを広域なネットワークテストベッドである JGN-X（Japan Gigabit Network 第 4 世代）上で実際に運用されているシステムに近い状況で検証していく．

現段階では，6. 1 節に示した SDN 環境の実験と同等な実験をこの FLARE 実機環境で行い，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

8.　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，実験開発により，Mininet 仮想環境と FLARE 実機環境の SDN 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．

今後は，ローカルの FLARE 実機上で DPN を生かした制御の検証を行った後，テストベット上での性能評価を行っていく．

今後の課題としては，DPN 環境や，さらに複雑なトポロジになった際に，この提案システムが正確に挙動するのか検証すると共に，より良い最適経路探索モデルを模索し，評価を行いたい．また，外部情報とコントローラをつなげる部分をどう設計していくかも今後の課題である．

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．

文　　献

[1] NTT ドコモ：「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

[2] 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

[3] 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」，DEIM2015，2015 年 3 月発表予定．

[4] 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.

[5] NEC：「OpenFlow の活用により災害時における移動通信ネットワークの動的なソフトウェア制御を実現」，http://jpn.nec.com/rd/innovation/feature/2013/11-openflow.html?，コンセンサス，2013 年 11-12 月号

[6] 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

[7] 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.

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

[9] Akihiro Nakao, “ Software-Defined Data Plane Enhancing SDN and NFV ”, Special Section on Quality of Diversifying Communication Networks and Services, IEICE Transactions on Communications , 2015 Jan

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

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