大規模災害時におけるSNS情報に基づいたアプリケーションQoS制御ライブラリの実装

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：東日本大震災時，電話やインターネットが使えない事態が生じたことから，大規模災害時には，従来のネットワーク機器監視による制御だけではなく新たな情報を利用した制御が期待されている．我々は，多くの人間の目や口コミによるネットワーク障害に対する集合知が災害の状況把握に有益であるという仮説に基づいて，SNS 情報に基づいたネットワーク制御システムを提案している [4]．従来の機器監視制御には，制御プレーンをプログラム可能とする SDN(Software Defined Networking) による柔軟な集中制御が期待されている．しかし，現在の SDN では，依然として，ヘッダー情報に基づく経路制御など従来と同様な制御の効率化に限定される．そこで我々は，データプレーン処理とその API(Southbound Interface) をプログラム可能とする SDN を拡張する DPN (Deeply Programmable Network) の概念 [7] に基づいて，SNS の情報が含まれるパケットペイロードのデータを監視することで経路制御をする手法を提案する．例えば，Twitter からリアルタイムにネットワーク障害を高い確度で検知し，ネットワーク全体の状況を迅速に把握することが可能になる．また，プログラム可能なデータプレーン処理を利用して，新たな情報源も動的に利用可能である．本論文では，FLARE[6] を用いて，OpenFlow の制御処理やアプリケーションの種類に基づいたトラフィックの分類を実装し，アプリケーション毎の QoS 制御を評価する．

1. はじめに

近年スマートフォンのような高機能なモバイル端末の普及とクラウドコンピューティングの発達などにより，インターネットを流れるトラフィックは日々増加し続けている．中でもビデオトラフィックは，今後 5 年で約 13 倍になるとも言われており，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].

このような近年のネットワークでは，利用集中により輻輳が発生する問題が考えられる．その最たる例が地震や台風などの大規模災害時で，トラフィックの急増に加え，ネットワーク機器の損壊など直接的な被害により輻輳が生じる．このような緊急時に，電話やメールなどの通信手段が利用可能であることは重要である．実際に 2011 年 3 月に発生した東日本大震災においても，一部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．この時，切り替え等の設定作業が一部手作業であったことと，ネットワーク全体の状態を迅速に把握できなかったことが原因のひとつとしてあげられた [2]．このことから，ネットワーク全体の状況を迅速に把握し，人手を介さない自動的なネットワーク制御システム構築の必要性が浮き彫りになった．

そこで，ネットワーク機器の集中管理・集中制御を可能にする SDN (Software Defined Network) や OpenFlow[3] といった技術を広域なネットワークに適用し，ネットワーク機器をソフトウェアで自動制御することが考えられてきた．しかし，ここで以下 2 つの課題が生じる．第一に，制御対象とするネットワークが広域な場合，障害の検知を，従来のようにネットワーク内部のモニタリング情報を基にして行うのでは難しいという課題である．第二に，ネットワークのプログラマビリティの限界である．SDN ではコントロールプレーンのプログラム可能性を実現するが，データプレーンのプログラム可能性は実現されない．よって，ネットワーク機器部分にはハードウェアが残り，可能な制御は，結果的に従来のネットワーク機器を用いたものと同等に留まるものが殆どとなる．つまり，有事に備えてソフトウェアによる柔軟な制御を取り入れる手法は，まだ研究開発の余地が大いにあると考えられる．

これらの背景に対し，本研究では，DPN (Deeply Programmable Network) 環境における SNS 情報に基づいたネットワーク制御システムを提案する．本提案システムは，

[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

緊急時に備えて最適な経路制御や帯域制御を自動的かつ柔軟に行う必要のあるネットワーク管理事業者を対象とし，大規模な緊急災害時でも，ユーザが安定してネットワークを利用できることを目的とする．

本提案システムの 2 つの大きな特徴を説明する．第一に，SNS 情報を利用して，経路制御を行うことである．我々は Twitter[5] からリアルタイムに障害を高い正確性で検知するシステムを構築した [4]．これは，ネットワークの障害をネットワーク内部ではなく，外部の集合知により検知するため，1 つ目の広域ネットワークでの障害検知の課題を解決する．この手法により，ネットワーク全体で障害の起こった地点を SNS 情報に基づき迅速に特定することができる．第二に，DPN 環境の適用である．DPN とは，コントロールプレーンだけでなくデータプレーンをもプログラム可能にするネットワーク仮想化の新たな概念である．これにより，ネットワークをフルにプログラム可能にし，2 つ目のネットワークのプログラマビリティの限界を解決する．このデータプレーンのプログラム可能性によって，アプリケーションの中身に基づく制御が可能となる．例えばアプリケーション毎のトラフィックを分別し，その各々に対する最適な QoS 制御である．大規模な災害時において，メールや電話，SNS のトラフィックを優先し，近年爆発的に増加した娯楽目的の動画配信のトラフィック帯域は制限するような優先制御手法は効果的であると判断できる．

本稿では，OpenFlow の機能とアプリケーションの中身に基づく制御機能の両方を実装可能な FLARE スイッチ [6]-[7] を用いて DPN 環境を構築し，この提案システムの実装と評価を行う．図 1 は提案システムを適用したネットワーク環境を示している．この図に示すように，大規模災害などの緊急時には [4] の手法により SNS 情報をサーバ上に集めて解析し，ネットワークの障害地点の特定を行う．この情報を基に，ネットワークコントローラで経路切り替えの判断を行い，ネットワーク上のスイッチに指示を送る．スイッチはネットワークの仮想スライスごとにアプリケーションを振り分け，アプリケーションの種類に応じた適切な帯域設定により，アプリケーション毎の QoS 制御を実現する．

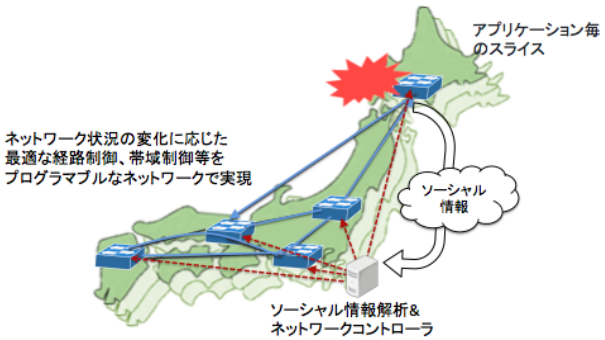

図 1　提案システムにより実現されるネットワーク

本論文の貢献は，以下の 2 点に集約される．

i　SNS 情報を利用したネットワーク障害検知システム [4] をネットワーク経路制御に統合した．これにより，SNS 情報に基づいて，ネットワークを自動的かつ自律的に制御できることを示した．

ii　DPN 環境の適用により，ネットワークを流れるパケットをアプリケーション別のトラフィックに振り分け，種類別に適切な帯域を設定するような，より高度なネットワーク制御が行えることを示した．

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 について説明する．次に，4 章で提案システムの概要を紹介する．そして，5 章で実験環境を示し，6 章で各実験について述べる．最後に，7 章で本稿をまとめる．

2. 関連研究

SDN や OpenFlow 技術を利用して，災害時に適切なネットワーク制御を自動的に行う障害対策手法が数多く存在する [8][9][10][11][12][13][14]．NTT ドコモら [8] は，OpenFlow 技術を用いて，低優先トラフィックに対する抑制を行うことで通信サービスの優先度に応じた制御を実現している．具体的には，DiffServ (Differentiated Services) を用いたクラスベースの CoS 制御で，受信パケットの通信フローとクラスのマーキングを行うことで，相対的に優先度を識別している．しかし，この優先制御は 1 人のユーザが複数の通信サービスを使用する状況下では，正常に識別可能でない．よって，1 ユーザが複数のアプリケーションを使用する場合でも正常に識別でき，各アプリケーションに対して各々に適切な制御を絶対的に行う本研究手法とは異なる．また，小川ら [9][10] は，IP アドレスと位置情報をマッピングして DB で管理し，DB と OpenFlow コントローラとの連携をとる手法を提案している．加えて，被災地の通信パケットに対して，IP ヘッダ中の ToS フィールドに災害 ID を付与することで被災地の通信パケットを識別し，QoS を実現する提案をしている．さらに OpenFlow コントローラに，緊急地震速報や気象情報などの外部災害情報を収集する機能を付加する点でも本研究と似ている．しかし本研究では，SNS による集合知を用いる点と，アプリケーションの種類毎に QoS 制御を行うという点で新規性がある．[11][12][13] については，無線アドホックネットワークに OpenFlow を適用するもので，ユーザが端末上で評価基準の重みを事前に登録し，それを基にした QoS 制御を提案している．江戸ら [14] は，災害リスクのモデル化を行い，モデル化された災害シナリオを基に経路制御を行っている．しかし，これらの研究はネットワークのトラフィックデータなど内部情報を基に通信制御を行おうとしており，SNS データという外部情報を基に通信制御を行う本研究とは異なる．

本研究では，SNS 情報の取得からアプリケーション毎の

QoS 制御までを自動的・自律的に行う．加えて，既存の研究はどれも，ソフトウェアコントローラによるネットワーク機器の集中管理という SDN の概念に留まっている．すなわちこれらの研究では，ネットワーク機器中で特別な制御をすることは考えられていない．この場合，通信内容に基づく詳細なレベルでの制御は難しい．よって本研究では，データプレーンのプログラム可能性を DPN 環境により適用し，自由に書き換え可能なソフトウェアスイッチを用いることで，アプリケーションの送信データの中身に基づいた詳細でより柔軟なネットワーク制御を行う．

3. SDN と DPN

3.1 既存の SDN/OpenFlow による制御

OpenFlow[3] は SDN を実現するためのプロトコルのひとつで，ネットワーク全体の集中管理・集中制御を可能にする．この技術は既に実用化段階で，主にクラウドを提供するデータセンターや企業内 LAN などの狭域ネットワークでの導入が進められている．

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**??**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**??**のように分離した．つまり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．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．

3.2 DPN/FLARE による制御

3.1 節より，SDN ではデータプレーンは，データ転送という固定的な動きしか実行しないことがわかる．このデータプレーンをもプログラム可能にするのが，DPN の概念である．つまり，DPN ではネットワークをフルにプログラム可能にする事を目的とする．

そして DPN を実現するために開発されたアーキテクチャの一つが，東京大学中尾研究室で研究中の FLARE 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．

このデータプレーン機能を FLARE では，Click module router[15] というソフトウエアルータで実現する．Click はスイッチやルータ等のネットワーク機器をソフトウェアで再現する言語である．Click の特徴は，「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．東京大学では，OpenFlow1.3 スイッチを Click のモジュールで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 では，この Click により作成された，プログラム可能な仮想スイッチをスリバーと呼ぶ．これらスリバーを物理ノード上に複数提供しそれらの集合で，スライスと呼ぶ仮想ネットワーク空間を作成する (図 2)．

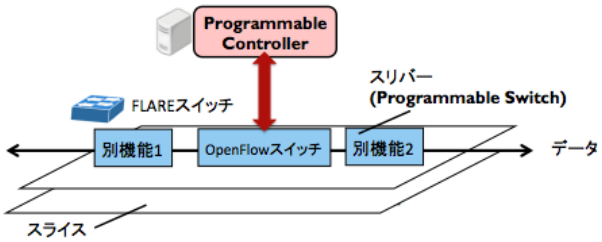

図 2　FLARE の仕組み

3.3 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．つまり，OpenFlow がネットワーク層までのデータを扱うのに対し，FLARE はアプリケーション層までのデータを扱うことができる．これによりアプリケーションの種類に基づく制御のような，より柔軟な制御を行うことが可能となる．本研究では SNS 情報に基づいた高度で柔軟なネットワーク制御を目指しているため，アプリケーション層のデータまで活用できる FLARE は最適なプラットフォームであると言える．

3.4 FLARE でのアプリケーションの弁別

FLARE でのアプリケーションの識別手法は複数存在する．そのうちのひとつである，文献 [16][17] を紹介する．この手法では，ホスト側のカーネルに変更を加え，アプリケーション名など制御に使いたい情報をトレイラとして付加することで識別する．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判別可能になる．これにより，アプリケーションの種別に基づく制御が容易に実現可能になる．

4. 提案システムの概要

本研究では，Twitter の解析から得られた外部障害情報をトリガとして，トラフィックの最適化をアプリケーション毎に自動的・自律的に行う，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 3 に示す．(1)-(4) についてはコントローラ上に Python で，(5) はスイッチ上に Click で実装し，一連の動作は全て自動化を行うものとする．動作は以下の通りである．

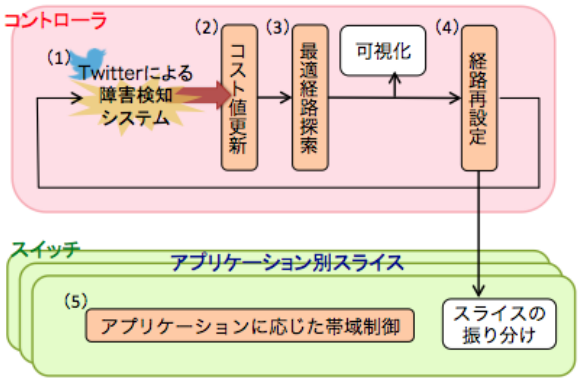

図 3　提案システム概要

(1) SNS の解析によるリアルタイム障害情報の受理
文献 [4] よりリアルタイムで Twitter を解析し，高い正確性でネットワーク障害を検知する．このシステムではツイートの本文や位置情報，時間情報を用いて，通信障害が発生している地域や原因，ユーザへの影響の度合いや復旧すべき地域の優先度などを算出する．それらの情報のうち，障害ツイート数と障害が発生した地域名，地域ごとの復旧の優先度を受理する．

(2) リンクのコスト値更新
Twitter から受理した情報を基に，地域間のリンクのコスト値を 60 秒間隔で更新する．例えば，デフォルトのコスト値を全て 1 に設定しておき，通信障害を検知した場合に以下の条件で更新する．障害ツイート中に対応させた地名を含むツイートが 20 件以上あれば，コスト値を 1 増加させる等である．つまりこれによって，コスト値が小さいということは，利用できる帯域が大きいことを表現する．

(3) 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．スタートからゴールへコスト値最小の経路が最適経路として設定させる．

(4) 経路再設定
フローエントリを REST-API を通してスイッチ側に投げることで，最適経路に経路を再設定する．

(5) アプリケーション毎の QoS 制御
文献 [16][17] よりアプリケーションの弁別を行う．弁別したアプリケーションは，種類別に各スライスに振り分ける．各スライスでは，Click の BandwidthRated-Splitter モジュールを使って，各アプリケーションに最適な帯域を設定する．これにより，アプリケーション毎の QoS 制御が実現される．

5. FLARE 実機環境

5.1 物理構成

本研究では，図 4 に示す物理構成で実機実験を行う．FLARE Switch1 と 2 にはそれぞれ，1G のポートが 8 個と 10G のポートが 2 個付いている．FLARE Switch3 と 4 にはそれぞれ，10G のポートが 4 個付いている．これら FLARE スイッチ 4 台を，1G と 10G の線を混合させメッシュ状に組む（図中で 1G は 1Gbps，10G は 10Gbps のネットワーク接続を示す）．図中の FLARE Central は FLARE 管理用のサーバで，このサーバで GUI によりスライスの作成等ができる．図 3 に示す提案システムの赤の部分であるコントローラを，この FLARE Central サーバ上に構築する．このコントローラから，4 台の FLARE スイッチを制御し，様々な制御モデルを検討する．各マシンの仕様は表 1 の通りである．

表 1　Specifications of machines

FLARE switch1 ～ FLARE switch4	Model	Sunwaytech SWS (barebone)
	CPU	Core i7-3612QE Mobile 2.1GHz
	Memory	8GB
	OS	CentOS 6.4
h1～h4	Model	Toshiba dynabook R734
	CPU	Core i5-4210 M 2.6GHz
	Memory	8GB
	HDD	SATA 500GB 5400RPM
	OS	Ubuntu14.04
h5, h6	Model	Dell PowerEdge R220
	CPU	Xeon E3-1241 v3 3.5GHz
	Memory	8GB
	HDD	SATA 1TB 7200RPM
	OS	Ubuntu14.04

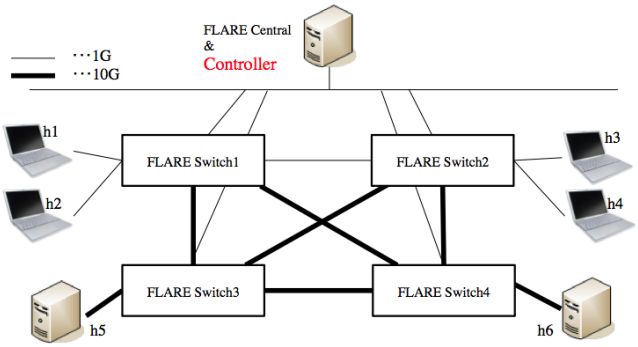

図 4　FLARE 物理構成

5.2 スループット測定

次に，前節で説明した FLARE 実験環境の動作確認と特性を把握するため，各経路のスループットを iPerf により測定した（図 4 に示していないトポロジを含む）．結果を

表 2 に示す．1G と 10G が混合している経路は，1G のみや 10G のみの経路に比べてややスループットの低下が見られる．このことより，1G と 10G の繋目がボトルネックになっていることが考察される．ただし，この測定された各経路のスループットは，次章で紹介する実験でのコスト値とは関係しない．

表 2　各経路のスループット測定結果

	経路	スループット
1G10G 混合	h1-s1-s3-h5	500(Mbits/sec)
	h1-s1-s2-s3-h5	590(Mbits/sec)
	h1-s1-s4-s3-h5	500(Mbits/sec)
	h1-s1-s2-s4-s3-h5	500(Mbits/sec)
1G のみ	h1-s1-h2	930(Mbits/sec)
	h1-s1-s2-h3	930(Mbits/sec)
10G のみ	h5-s3-h6	2.1(Gbits/sec)
	h5-s3-s4-h6	1.7(Gbits/sec)

6. FLARE 実機実験

6.1 実験内容

行う実験は以下の 3 つである．

- 提案システムの動作確認実験
- スループット評価実験
- 遅延時間（RTT）評価実験

初めに提案システムの動作を確認する実験を行う．次にこのシステムを評価するため，スループット測定実験を行う．最後に，経路切替にかかる遅延時間を見るために，遅延時間測定実験を行う．

6.2 提案システムの動作確認実験

本研究では，提案システムを広域なネットワークテストベッド上で動作させることを想定しているが，本実験では便宜的に，前章で示した FLARE 実験環境でエミュレーションを行う．本節では，提案システムの挙動を実行例とともに詳しく説明しつつ，動作の確認実験を行う．加えて，正しく動作することを示し，結果の可視化を共に示す．

6.2.1 提案システム実行例

図 3 に示す提案システム中の (1)-(4) について動作を確認する．(5) については，OpenFlow1.3 を実現する Ofswitch モジュールを実装した Click プログラムを動作させる．これにより，FLARE のプログラミング環境の上に OpenFlow を実装した，SDN レベルの実験を行う．本実験では，東日本大震災時の 2011 年 3 月 11 日 14 時から 15 時の実際のツイートを用いる．FLARE スイッチを 1 から順に，岩手，京都，東京，福岡の地域名に対応づけ，通信としては岩手付近の h1 からスタートし，東京付近の h5 をゴールとして設定する．本実験により，災害時に Twitter 情報に基づいて，障害地区を迂回する経路切り替え制御を，アプリケーション毎の QoS を考慮して行う，提案システムの動作が確認できる．

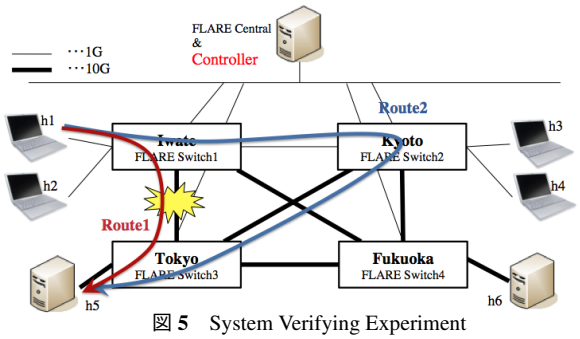

図 5　System Verifying Experiment

まず全てのリンク間のコスト値はデフォルトで 1 なので，コスト値最小の Route1 の経路が設定される．コスト値の更新は 60 秒間隔で行われ，Twitter の解析システムより岩手東京間で障害が検知されたことによって，2 回目の 120 秒後の更新の際に岩手と東京間のコスト値が＋ 1 に更新される．その後も 60 秒間隔で岩手東京間で障害が検知され続けたことにより，Route1 の経路はコスト値が＋ 1 され続ける．3 回目の 180 秒後の更新の時点で，岩手と東京間のコスト値が 3 にるため，コスト値最小の 2 である京都を通る経路 Route2 が選択される．ダイクストラ法によって Route2 が選択されると，Route2 に経路を設定する REST-API が呼び出される．これにより，フローエントリがスイッチ側に投げられて，経路が再設定される．

6.2.2 スイッチのポート番号による経路切替確認

前節に示した実験より，経路切替に成功し，実際に経路が切り替わったことが確認できるキャプチャ画面を図 6 に示す．画面中の”538247169”から小さい順に FLARE Switch1～4 を示す．輻輳の前後を表す青い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 Route1 から Route2 へ切り替わったことが確認できる．

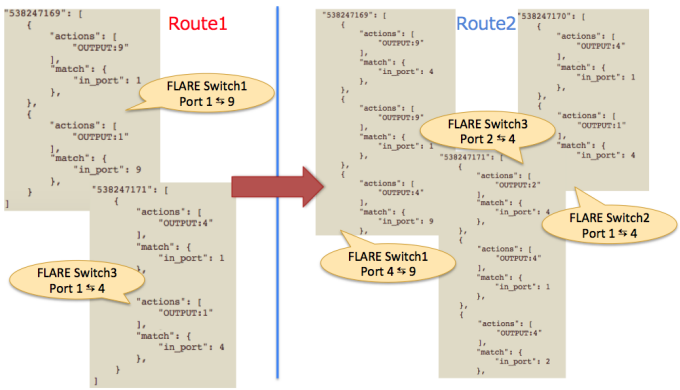

図 6　障害イベントによる自動経路切替えの確認

6.2.3 可視化

図 3 の提案システム中に示す，可視化の出力結果を図 7 に示す．(3) より出力された最適経路のリストを基に Google Map 上に可視化を行う．Route1 から Route2 への経路切替が GUI 上でも確認できる．

図 7　経路可視化 web サイト

6.3　スループットの測定実験

次に，本提案システムを評価するため，iPerf を用いたスループット測定実験を行う．図 8 は，提案システムを適応させた場合と適応させていない場合の Route2 のスループット性能の差を示したものである．ほとんど差がないことから，本提案システムは経路切り替えのオーバーヘッドなく，ハードウェア本来の性能を出せることがわかる．

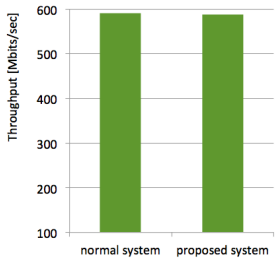

図 8　スループットの比較

6.4　遅延時間の測定実験

次に，経路切替にかかる時間を把握するため，Ping を 1ms 秒毎に 1 つ飛ばし遅延時間を測定する．結果を図 9 に示す．経路切替時に 20-30ms 程度の遅延が発生しているのは，コントローラにフローエントリを問い合わせるためである．また，経路切替時でない時にでも小さい遅延が多数発生しているのがわかる．これはボトルネックになっていると考えられる 1G と 10G の繋目の部分で発生するものだと考察される．

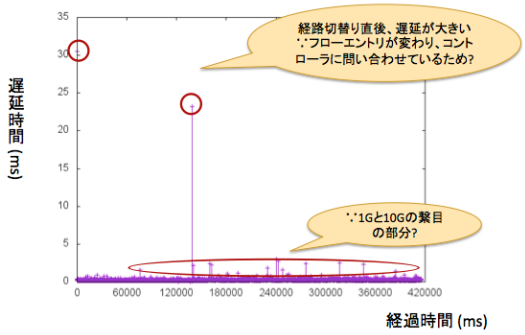

図 9　往復遅延時間（RTT）

7.　まとめと今後の課題

本論文では，SNS によるリアルタイム障害情報に基づいて，アプリケーションの中身を考慮する高度で柔軟なプログラマブル経路制御手法を提案する．本論文の貢献は以下の通りである．

第一に，Twitter 解析システムからのインプットより，自動的に障害地区を迂回するシステムの構築である．FLARE4 台を用いた実験より，経路切替のオーバーヘッドなくハードウェア本来の性能を出せることに加え，20-30ms 程度の RTT 値で経路切替が出来ることが確認できた．よって，FLARE7 台で実装される広域ネットワーク JGN-X[18] 上でも提案システムが充分な性能で動作可能であることが導かれる．

第二に，アプリケーション毎の振り分けと QoS 制御の設計と試作である．アプリケーション毎のスライス制御が確認できたことから，アプリケーション毎の QoS を含めた提案システム全体の性能評価を今後行う．

今後の課題としては，Twitter からユーザ側の状況をより詳細に抽出し，それを反映したユニークなネットワーク制御を行うことである．これにより，さらに高度できめ細やかなネットワーク制御を行う．

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および科学技術振興機構戦略的創造研究推進事業（CREST）によるものである．

参考文献

[1] Cisco Visual Networking Index(VNI), ”Global Mobile Data Traffic Forecast Update, 2014-2019”, White Paper, http://www.cisco.com/web/JP/solution/isp/ipngn/literature/pdf/white_paper_c11-520862.pdf, Feb. 2015.

[2] NTT DOCOMO, ”Improvement of Credibility for Operation System in the Case of Large Disaster”, https://www.nttdocomo.co.jp/binary/pdf/corporate/technology/rd/technical_journal/bn/vol20_4/vol20_4_026jp.pdf, Technical Journal Vol. 20 No. 4, 2013.

[3] N.McKeown, T.Anderson, H.Balakrishnan, G.Parulkar, L.Peterson, J.Lexford, S.Shenker, and J.Turner. ”OpenFlow: enabling innovation in campus networks.” ACM SIGCOMM Computer Communication Review, Vol.38, No.2, pp.69-74, 2008.

[4] Chihiro Maru et. al., ”Network Failure Detection System for Traffic Control using Social Information in Large-Scale Disasters”, ITU Kaleidoscope Conference 2015: Trust in the Information Society, S5.3, Dec. 2015.

[5] Twitter, http://twitter.com/

[6] Akihiro Nakao, ”FLARE: Open Deeply Programmable Network Node Architecture,” Stanford Univ. Networking Seminar, October 2012. http://netseminar.stanford.edu/10_18_12.html

[7] 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, pp.12-19, Jan. 2015.

[8] NTT ドコモ，東北大学，日本電気株式会社，富士通株式会社，株式会社日立ソリューションズ東日本：「大規模災

害時における移動通信ネットワーク動的通信制御技術の研究開発」総務省平成 23-24 年度研究開発.
[9] 小川康一，吉浦紀晃：「災害 ID 付与方式による災害時のネットワーク優先配送 OpenFlow による実装と評価」, 電子情報通信学会技術報告, vol.2014-IOT-24, no.23, pp.1-6, 2014-02-20
[10] 小川康一，吉浦紀晃：「通信の信頼性確保を考慮した位置情報に基づくネットワーク運用手法」,DICOMO 2015, 8B-2, pp.1646-1652, 2015
[11] 熊谷友来，関野雄人，内田法彦，柴田義孝：「OpenFlow をベースとした災害時における End-to-End 通信路の選択方法の実現」, 情報処理学会第 75 回全国大会, pp.3-337-338，2013 年 3 月.
[12] 関野雄人，柴田義孝，内田法彦，白鳥則郎：「OpenFlow をベースとした災害情報ネットワークにおけるリンク切り替え技法の実現に関する研究」, 情報処理学会 SIG, Vol.2013-DPS-154 No.49，2013 年 3 月.
[13] 佐藤剛至，柴田義孝，内田法彦：「SDN によるコグニティブ無線技術を基盤とした災害に強いネバー・ダイ・ネットワークに関する研究」, マルチメディア通信と分散処理, IPSJ-DPSWS2013006, pp.46-52，2013 年 12 月.
[14] 江戸麻人，和泉諭，阿部亭，菅沼拓夫：「災害リスクを考慮したスマートルーティングの設計と実装」, DICOMO 2015, 7E-3, pp.1520-1524，2015 年 7 月.
[15] The Click Modular Router Project : http://www.read.cs.ucla.edu/click/
[16] Akihiro Nakao, Ping Du. ”Application and Device Specific Slicing for MVNO”, 2014 International Science and Technology Conference (Modern Networking Technologies) (MoNeTeC), Oct. 2014.
[17] Akihiro Nakao, Ping Du, Takamitsu Iwai. ”Application Specific Slicing for MVNO through Software-Defined Data Plane Enhancing SDN”, IEICE TRANSACTIONS on Communications Vol.E98-B, No.11 pp.2111-2120, 2015.
[18] JGN-X, http://www.jgn.nict.go.jp