Vol.2016-IOT-32 No.13
2016/3/3

耐障害性・耐災害性の検証・評価・反映プラットフォームの設計と実装

北口 善明[1,a)]　柏崎 礼生[2]　近堂 徹[3]　市川 昊平[4]　西内 一馬[5]　中川 郁夫[2,6]　菊池 豊[7]

概要：広域分散システムの耐災害性や耐障害性が十分なものであるか評価する仕組みとして，我々はソフトウェア制御により災害や障害を模倣することで検証を行う障害発生プラットフォームを提案している．本稿では，SDN 技術として RESTCONF API によりネットワーク機器を操作する OpenDaylight フレームワークを採用して構築した検証プラットフォームの設計と実装について述べる．

キーワード：耐障害性, 故障耐性, SDN, OpenDaylight

A design and implementation of a verification and evaluation platform for failure resistance and failure tolerance

Yoshiaki Kitaguchi[1,a)]　Hiroki Kashiwazaki[2]　Tohru Kondo[3]　Kohei Ichikawa[4]
Kazuma Nishiuchi[5]　Ikuo Nakagawa[2,6]　Yutaka Kikuchi[7]

Abstract: We propose a platform to evaluate disaster tolerance and fault resistance of wire-area distributed systems with an SDN technology emulating artificial disasters and faults. In this report, we introduce a design and implementation of the platform using RESTCONF APIs of the OpenDaylight frameworks as SDN technology.

Keywords: Failure resistance, Failure tolerance, SDN, OpenDaylight

1. はじめに

ICT システムは市民生活から基幹業務に至るまで情報化社会を支えるシステムとして必要不可欠なものとなり，システムの堅牢性も重要な要素のひとつとなっている．そのような要求に対して，冗長性を確保することや，広域分散により単一障害点を回避といった手法が考えられる．しかしながら，東南海大地震等の災害では同時多発的に障害が発生し，通常想定する範囲内に収まらない障害を想定する必要がある．

広域分散システムの基盤となるインターネット網において，災害時においてもインターネットへの接続性を持続するための研究や活動として，衛星通信による域外通信の確保，自組織資源の解放と相互接続によるネットワーク再構成，手持ちの機材を用いたネットワーク構成訓練などが行われている．これらは災害の発生後，被災者救援や支援，復興といった次のステップに繋がる最低限の ICT 環境を構成するための研究・活動である．

防災の観点では，広域分散システムには単に平常時と全く同じ機能を実現するような堅牢性だけでなく，多発的な障害が発生しつつも可能な限りの品質で通信を維持できるようなしなやかさ（レジリエンス）や，生き残っている通信サブシステムの合成による再構成を柔軟に可能にするこ

[1] 金沢大学 総合メディア基盤センター
Kakuma-machi, Kanazawa, Ishikawa 920-1192, Japan
[2] 大阪大学サイバーメディアセンター
[3] 広島大学情報メディア教育研究センター
[4] 奈良先端科学技術大学院大学情報科学研究科
[5] 株式会社シティネット
[6] 株式会社インテック
[7] 高知工科大学地域連携機構
[a)] kitaguchi@imc.kanazawa-u.ac.jp

表 1 ネットワーク障害の分類
Table 1 Classification of network failure

発生区分	障害要因	具体的な症状	実装する機能
制御・運用・ソフトウェア	通信規制制御	輻輳	遅延発生＋ N%パケットロス
			トラフィックシェーピング
	不正な経路伝播	ルーティングループ	経路表強制書き換え ※
		ルートフラップ	
		宛先不達（経路障害）	
ネットワーク機器	装置故障（全体）	通信断（全体）	インタフェースダウン ※
	装置故障（部分）	通信断（部分）	
	リソース過負荷	パケットロス	N%パケットロス
		遅延	遅延発生
通信回線	拠点間通信ケーブル断	通信断	インタフェースダウン ※，100%パケットロス ※
	中継機・交換機故障		
	トラフィック集中	輻輳	遅延発生＋ N%パケットロス
			トラフィックシェーピング
設備環境	局舎損壊（浸水・火災等を含む）	通信断（全体）	インタフェースダウン ※，100%パケットロス ※
	電源喪失		
	空調故障		
		通信断（部分）	

※ プロトタイプ実装した機能

と（フレキシビリティ）が求められている．さらにこれは技術的に実現可能というだけではなく，災害の現実に近い状況での評価により，実用的に機能するかを検証することが可能でなくてはならない．

我々は，これらの要求を受け，障害や故障を模倣することで情報システムにおける耐障害性を評価する，耐障害性・耐災害性の検証・評価・反映プラットフォーム（以下，障害発生プラットフォーム）を提案してきた [1]．これまでの実装では，提案する障害機能をすべて実現することができておらず，障害実装においてもベンダ固有の機能を利用していることにより汎用性に欠ける問題があった．本稿では，ネットワーク機器に対して発生させる障害実装をより汎用的にするため，OpenDaylight フレームワークを活用した障害発生プラットフォームの設計と実装について報告する．

2. 障害発生プラットフォームの機能

我々が提案する障害生成プラットフォームでは，自然災害等による被災状況を検証環境上にて模倣し，広域分散システムの耐災害性・耐障害性を検証・評価することを目的としている．このプラットフォームは以下の 3 つのフレームワークにより構成される．

- 故障生成フレームワーク
- 障害発生フレームワーク
- 状態収集フレームワーク

以下に，これらのフレームワークについて解説する．

2.1 故障生成フレームワーク

故障生成フレームワークは災害による障害が「いつ，どこで，何が」起こるかが表現される「災害シナリオ」を記述するためのフレームワークである．災害シナリオは，過去に発生した災害と，これから発生することが危惧される災害とに分割することができる．過去に発生した災害においては「いつ，どこで，何が」を把握できるため，これら 3 つの情報を記述することができるマークアップ言語を用いてシナリオを作成する．

既に起こった災害時にどの箇所で障害が発生したかという情報は，回線事業者によって公開ポリシーが異なるものの，例えば国立情報学研究所 (NII) が提供する学術情報ネットワークである SINET4 では，東日本大震災時の回線状況について公開している [2]．この公開情報によると，仙台〜金沢間および仙台〜東京間では回線の正副両系の切断が生じ不通になり，また仙台 DC は発電により 96 時間サービスを持続した，とされている．さらに詳細な時系列についても照会すれば情報を獲得できることが期待でき，この時系列に沿った障害発生情報をもとに東日本大震災時の SINET4 のネットワーク障害を模倣することができる．この障害発生情報をマークアップ言語により記述し，統合することで広域の障害を再現するフレームワークとなる．

2.2 障害発生フレームワーク

障害発生フレームワークは，前述の故障生成フレームワークで作られた災害発生シナリオに記述された「何が」をネットワーク機器に対して実現するフレームワークである．我々は，総務省「大規模災害等の緊急事態における通信確保の在り方に関する検討会」で示されている災害事象 [3] や「情報通信ネットワーク安全・信頼性基準」の内容 [4] をもとに，災害時における通信設備等に対する障害に焦点を絞り，各障害事象に対してフレームワークとして必要となる制御について検討した [1]．表 1 はこれらを細

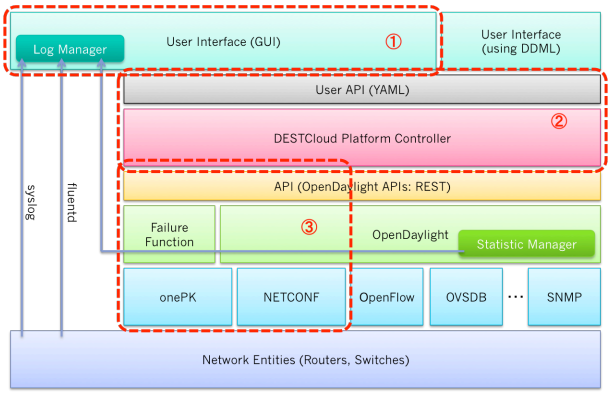

図 1 障害発生プラットフォームのアーキテクチャ
Fig. 1 Architecture of the failure emulation platform

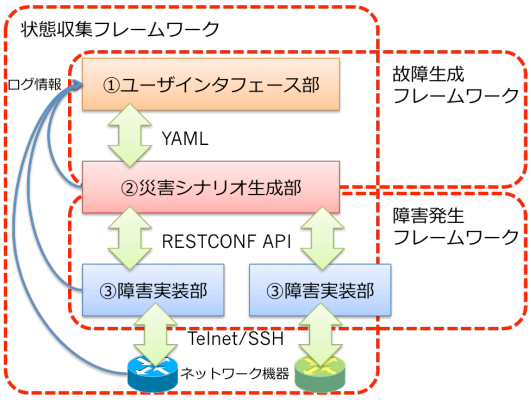

図 2 各フレームワークと機能部の関係
Fig. 2 Relationship of each frameworks and the functions

かな区分に分類し，それぞれの障害要因と具体的な症状の対応付けを行い，具体的な症状を模倣するために本フレームワークで実現する機能についてまとめたものである．

表 1 に示す「実装する機能」のうち，※印を付与したものがプロトタイプとして実装し評価を行ってきた項目である．障害発生フレームワークのプロトタイプ実装には，SDN（Software Defined Network）技術を用いることで，プログラマブルな障害発生の有効性を検証できている [1]．ただ，SDN プラットフォームとして利用した Cisco Systems 社による onePK（One Platform Kit）では，通信遅延の挿入やパケットロスの発生が難しいという課題があり，また，幅広いネットワーク機器で利用できない制限が問題であった．これらの課題を解決する必要がある．

2.3 状態収集フレームワーク

状態収集フレームワークでは，障害発生フレームワークにより引き起こされた障害により，ネットワーク機器や評価する広域分散システムがどのような状態であるかを収集し統合的に扱うフレームワークである．実環境における手動でのネットワーク防災訓練では，障害発生の前後で通信システム全体がどのような状況に置かれていたのかを知るためのログ等の収集が膨大かつ迅速に行うことができず，十分な検証をすることが難しいとの報告がある [5]．障害発生の前後において通信システム全体の状態把握を行うことは，検証対象のシステムの動作検証を行う上で必要であり，本フレームワークにおいて様々なログ情報を統合的かつ迅速に収集する．

本フレームワークで収集したログ情報は，障害発生プラットフォームの操作画面（GUI）において統合的に閲覧できる機能を提供する．収集したログデータは時系列やネットワーク機器毎に参照することを可能とし，利用した災害シナリオと比較することで，検証・評価を的確に行えることを実現する．

3. 障害発生プラットフォームの設計

前節で提示したフレームワークを実装するため，実装する障害発生プラットフォームのアーキテクチャを図 1 と図 2 のように設計した．以下において，図中に示した以下の 3 つの機能部についてそれぞれ解説する．

① ユーザインタフェース部
② 災害シナリオ生成部
③ 障害実装部

3.1 ユーザインタフェース部

検証作業を行う利用者が障害発生プラットフォームを操作するために，GUI による「ユーザインタフェース部」を用意する．また，このユーザインタフェースでは，障害発生プラットフォームの管理者により障害発生プラットフォームが提供するネットワーク機器により構成されるトポロジ（以下，全体トポロジ）の設定作業にも利用される．

ユーザインタフェース部での入力情報を災害シナリオ生成部に伝達するための API では，構造化されたデータを表現するためのフォーマットである YAML（YAML Ain't Markup Language）*1 を用いることとした．表現するデータ形式として「エンティティ YAML」「トポロジ YAML」「災害シナリオ YAML」を定義する．

以下に，利用者が検証作業を行うに際し，ユーザインタフェース部にて行われる手順を示し説明する．

(1) ネットワーク機器情報の登録（管理者）
障害発生プラットフォームで利用するネットワーク機器情報を管理者が登録する．ユーザインタフェース部で入力されたネットワーク機器の情報は，エンティティ YAML 形式として災害シナリオ生成部に渡される．エンティティ YAML が登録されると，登録されたネットワーク機器から情報を収集し，全体トポロジが災害シナリオ生成部から提示される．

*1 http://yaml.org/

(2) 検証環境ネットワークトポロジの設計（利用者）
利用者は，障害発生プラットフォーム上に構築する検証用のネットワークトポロジ（以下，サブトポロジ）を決定する．このサブトポロジは，全体トポロジを元に不要なネットワーク機器やリンクを削除することで作成する．作成されたサブトポロジ情報はトポロジ YAML として災害シナリオ生成部に送られる．

(3) 災害シナリオの作成（利用者）
利用者は，サブトポロジ上に構築した広域分散システムの検証に用いる災害シナリオを作成する．障害情報はマクロなもの（エリアダウンや拠点障害など）やミクロなもの（機器障害や回線障害など）を用意し，時系列に選択可能とする．合わせて，故障生成フレームワークとして，これまでに発生した災害に基づいた災害シナリオを生成する機能もここで実現する．

(4) 災害シナリオによる検証作業（利用者）
作成した災害シナリオは災害シナリオ YAML として構成し，検証作業時に災害シナリオ生成部に送られ，時系列に発生される障害により，対象システムの耐障害性検証が行われる．災害シナリオの実行が完了すると初期状態へと回復し，ユーザインタフェース部で収集するログ情報を元にした評価作業を可能とする．

前述したように，ユーザインタフェース部では，障害発生プラットフォームの操作に加えログマネージャ機能を有し，状態収集フレームワークの操作も可能とする．機器のログ情報収集には，一般的な Syslog や fluentd に加え，障害実装部で用いる OpenDaylight の機能（Statistic Manager）の利用も想定する（3.3 節にて議論）．

3.2 災害シナリオ生成部

災害シナリオ生成部は，障害発生プラットフォームの核となる部分であり，故障生成フレームワークと障害発生フレームワークの連携部となる．ユーザインタフェース部からの YAML データを元に，ネットワーク機器の制御を実現する．このネットワーク機器制御には，オープンソースで開発が行われている SDN コントローラである OpenDaylight[*2] フレームワークを利用する．OpenDaylight では，ネットワーク機器の制御プロトコルの差異を吸収し，汎用的な RESTCONF API の提供を可能とするフレームワークを提供している．我々もこれに習い，RESTCONF API を障害実装部にて実現し，災害シナリオ部からの制御を行う設計とした．

また，ユーザインタフェース部から指定される障害には，ネットワーク機器への障害以外にも回線への障害が存在している．OpenDaylight フレームワークでは，ネットワーク機器への制御となるため，回線に対する障害についてはその両端のネットワーク機器への制御に分解して障害生成を行う．

以下に，災害シナリオ部における主な機能に記し，それぞれ説明する．

- ネットワーク機器情報管理
  エンティティ YAML で登録された情報を DB にて管理する．
- トポロジ情報管理
  全体トポロジと合わせて，検証時に登録されるサブトポロジの管理を行う．サブトポロジは災害シナリオ生成部にて ID が設定される．
- 災害シナリオを元にした機器制御
  災害シナリオに記載された情報から，実際に操作が必要なネットワーク機器を算出し，必要な障害発生処理をネットワーク機器に対して実施する．先に述べたように，ネットワーク機器制御には OpenDaylight フレームワークを用いる．

また，災害シナリオ生成部では，障害発生プラットフォームにおけるログ情報生成も担っており，災害シナリオによるネットワーク機器に対する制御情報を記録し，ログマネージャに対して通知する．

3.3 障害実装部

障害実装部は，利用するネットワーク機器の差異を吸収する目的で用意する．実装する機能は表 1 に挙げたものとし，すべての障害機能を実現する．

今回利用する OpenDaylight フレームワークでは，このネットワーク機器における差異を吸収した標準的な制御インタフェースを提供することを目指しているが，現状の実装では機種依存性を吸収するような作りとはなっていないことが確認できた．そのため，ネットワーク機器への制御を中継する障害実装部を用意し，OpenDaylight の思想を継承した機種依存性のない API を災害シナリオ生成部に対して提供する．そのため，OpenDaylight の一部である Statistic Manager は今回の実装では利用しないこととし，ネットワーク機器から syslog で状態収集する実装とした．

また，ネットワーク機器制御プロトコルには YANG データモデル [6] を用いた NETCONF[7] を想定する．YANG（Yet Another Markup Language）データモデルは，ネットワーク機器の構造や各種設置値などを抽象化し，ネットワーク機器の差異を吸収し汎用的に扱うことを目的としたデータモデルである．YANG による各種データのモデル化は，IETF における様々な WG にて盛んに議論されている過程であり，今後の標準的な技術になると考えられる．我々も障害実装部で持つデータを YANG データモデルにて設計する．

[*2] https://www.opendaylight.org/

```
- entity_name: Osaka MX80
  ID: EE36120B-F056-436A-A2A0-0E0EC2BB68E7
  IP_address: 192.168.100.1/24
  port_number: 22
  protocol: ssh
  network_os: junos
  username: admin
  password: foobar
-entity_name: Hana VyOS
  ID: BE379CC6-D658-42F5-AE1E-3050F20DC5F8
  .....
```

図 3 YAML データ形式例（エンティティ YAML）

Fig. 3 Example of YAML data format (Entities YAML)

```
- entity_type: router
  entity_name: Osaka MX80
  ID: EE36120B-F056-436A-A2A0-0E0EC2BB68E7
  IF_numbers: 4
- entity_type: router
  entity_name: Naha VyOS
  ID: BE379CC6-D658-42F5-AE1E-3050F20DC5F8
  IF_numbers: 2
- entity_type: router
  entity_name: Kanazawa Juniper MX80
  .....
- entity_type: line
  entity_name: Osaka-Naha
  ID: B00BE291-F1C5-496F-9E1F-A62280D257BB
  nodes:
    - routerID: EE36120B-F056-436A-A2A0-0E0EC2BB68E7
      router_IF_number: 3
    - routerID: BE379CC6-D658-42F5-AE1E-3050F20DC5F8
      router_IF_number: 0
- entity_type: line
  entity_name: Osaka-Kanazawa
  .....
```

図 4 YAML データ形式例（トポロジ YAML）

Fig. 4 Example of YAML data format (Topology YAML)

```
- fault_type: port down
  fault_ID: 4FD63718-4014-4140-AE44-30F2034FB0BA
  time: 0
  fault_place:
    router_ID: EE36120B-F056-436A-A2A0-0E0EC2BB68E7
    interface_number: 3
- fault_type: packet loss
  fault_ID: 7486A59E-7530-44DF-BB4A-E946CA8792A5
  time: 4000
  fault_place:
    line_ID: B00BE291-F1C5-496F-9E1F-A62280D257BB
    loss_ratio: 50
- fault_type: traffic shaping
  fault_ID: 63AB14FD-42CB-438B-954B-E430D357E424
  time: 4000
  fault_place:
    line_ID: 06769A4C-90FD-480E-B903-7823296DE3FC
    bandwidth: 50
- fault_type: recover
  fault_ID: 63AB14FD-42CB-438B-954B-E430D357E424
  time: 10000
- fault_type: recover
  .....
```

図 5 YAML データ形式例（災害シナリオ YAML）

Fig. 5 Example of YAML data format (Disaster Scenario YAML)

4. 障害発生プラットフォームの実装

前章での設計を元に，３つの機能部に分けて障害発生プラットフォームを実装した．以下に各構成部間の連携部分に焦点を当てて，実装の詳細を記述する．

4.1 YAML フォーマットの定義と実装

ユーザインタフェース部と災害シナリオ生成部との間で用いる YAML データを，先に示したように３つの形式で定義した．図 3，図 4 および図 5 にそれぞれの YAML データ形式の実例を示す．

```
grouping fault-action {
    leaf interface { type string; }
    container action {
        leaf port-down { type empty; }
        leaf all-lose { type empty; }
        leaf packet-loss {
            type uint8 { range “1..99” ; }
        }
        container route-change {
            leaf prefix { type inet:ipv4-prefix; }
            leaf nexthop { type inet:ipv4-address; }
        }
        leaf shaping { type uint32; }
        leaf delay { type uint16; }
        uses direction-ref;
    }
}
```

図 6 YANG データモデル（障害生成情報）

Fig. 6 YANG data model (for fault action)

```
rpc get-interface {
    input {
    }
    output {
        list interface {
            key "name";
            leaf name { type string; }
            uses interface;
        }
    }
}
rpc fault-action {
    input {
        uses fault-action;
    }
    output {
        leaf result { type boolean; }
        leaf message { type string; }
        leaf uuid { type string; }
    }
}
rpc recovery {
    input {
        leaf uuid { type string; }
    }
    output {
        leaf result { type boolean; }
        leaf message { type string; }
    }
}
rpc reload {
    input {
    }
    output {
        leaf result { type boolean; }
        leaf message { type string; }
        leaf-list uuid { type string; }
    }
}
```

図 7 YANG データモデル（RPC）

Fig. 7 YANG data model (RPC)

ユーザインタフェース部で入力されたエンティティ YAML には，ネットワーク機器毎に ID が設定され，この情報を元に，災害シナリオ生成部が障害実装部を介してネットワーク機器にインタフェース情報を問い合わせる．災害シナリオ生成部において，ネットワーク機器からのインタフェース情報を元に機器間の接続性を計算し，全体トポロジをトポロジ YAML によりユーザインタフェース部に通知する．図 4 中の回線（line）に対する ID は，この時に災害シナリオ生成部にて付与する．

災害シナリオ YAML では，発生させる障害毎に ID を設定し，時系列で制御を指定する．図 5 の実例のように，障害からの回復には対象とする事象の ID を指定することで，発生中の障害情報を管理しながら災害シナリオを遂行する．

4.2 RESTCONF API の定義と実装

災害シナリオ生成部と障害実装部との間では RESTCONF API を利用する．この RESTCONF API では，YANG データモデルにて定義しており，以下のデータモデルを定義した．

- ネットワーク機器情報（管理用）
- 障害生成情報（管理用）
- インタフェース情報

定義した YANG データモデルの一部を図 6 および図 7 に示す．RESTCONF API では，表 1 で示した障害機能のための RPC（Remote Procedure Call）に加え，ネットワーク機器のインタフェース情報を返す RPC を実装した．発生させた障害に対して障害実装部にて ID を割り当て，障害を回復するためにはこの ID を要求することで，障害発生状態を障害実装部にて管理する．災害シナリオの終了イベントの際には，障害実装部にて保持する状態を元に初期設定に戻す実装としている．

4.3 JGN-X を用いた全体トポロジ

障害発生プラットフォームで用いるネットワーク機器には，OpenDaylight プロジェクトメンバとして参画している Juniper Networks 社と Brocade Communications Systems 社の機器を検討した．ただ，両社の実装において表 1 で定義した障害機能の調査を進めた結果，「遅延発生」と「N%パケットロス」の実装がルータ単体の機能で実現できないことが確認できた．

そこで，これらの機能を実現できるネットワーク機器の選定を進め，今回はソフトウェアルータである VyOS[*3] でのみすべての障害機能を実現することとした．合わせて，「遅延発生」と「N%パケットロス」を除く実装を，JGN-X[*4] の IP 仮想化サービス [8] にて利用可能な Juniper Networks 社の MX80 に対して適用した．

図 8 に，JGN-X 上に構築した障害発生プラットフォームにおける全体トポロジを示す．今回用意した拠点は大阪拠点を中心に札幌，金沢，奈良，高知，広島，那覇の合計 7 拠点で，各拠点は 3 拠点とそれぞれ直接接続する構成としている．各拠点におけるルータには，札幌と那覇拠点に VyOS を用意し，残りの拠点は MX80 としている．したがって，「遅延発生」と「N%パケットロス」の機能は札幌と那覇拠点にのみ適用可能となっている．

5. おわりに

広域分散システムの耐災害性や耐障害性を検証する仕組みとして障害発生プラットフォームを提案し，実装した．実際にネットワーク機器を制御し災害や障害を模倣することで，広域分散システムの冗長性や堅牢性の評価が可能となる．今回の実装では，ネットワーク機器制御に関してより汎用性を持たせるために，OpenDaylight フレームワークを利用したが，汎用性の面で課題が残る結果であった．

今後，開発したプラットフォームを用いた検証・評価を進め，提案手法の有用性を示す予定である．

[*3] http://vyos.net/wiki/Main_Page
[*4] http://www.jgn.nict.go.jp/

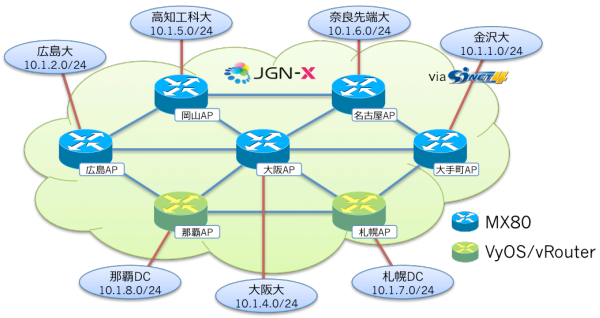

図 8 JGN-X 上に構築した全体トポロジ
Fig. 8 Entities topology built on JGN-X

謝辞 本研究は，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「分散システムの耐災害性・耐障害性の検証・評価・反映を行うプラットフォームとビジネスモデルの開発 (150202001)」の助成を受けた．また，日本学術振興会産学協力研究委員会インターネット技術第 163 委員会（ITRC）地域間インタークラウド分科会（RICC），コンピュータリソースの提供をいただいた各大学，JGN-X の回線を提供いただいた情報通信研究機構に深謝する．

参考文献

[1] 北口善明, 柏崎礼生, 近堂徹, 市川昊平, 西内一馬, 中川郁夫, 菊池豊. 広域分散システムの耐障害性を評価する検証プラットフォームの実装と評価. to appear in 情報処理学会論文誌, Vol. 57, No. 3, March 2016.
[2] 国立情報学研究所. 東日本大震災でもサービスの提供を続けていた SINET4. In *NII Today*, 第 52 巻, pp. 8–9. 国立情報学研究所, 2011.
[3] 総務省. 大規模災害等緊急事態における通信確保の在り方に関する検討会. http://www.soumu.go.jp/main_sosiki/kenkyu/saigai/ (参照 2015-01-30).
[4] 総務省. 情報通信ネットワーク安全・信頼性基準. http://www.soumu.go.jp/manu_seisaku/ictseisaku/net_anzen/anshin/ (参照 2015-01-30).
[5] 岡村健志, 菊池豊, 福本昌弘, 豊永昌彦, 佐々木正人, 今井一雅, 山田覚, 風間裕, 一色健司, 名和真一, 高畑貴志. 地域 IX における人為的障害による耐災害性の検証. マルチメディア, 分散, 協調とモバイル (DICOMO2014) シンポジウム, pp. 485–489. 情報処理学会, July 2014.
[6] M. Bjorklund. YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF). RFC 6020 (Proposed Standard), October 2010.
[7] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman. Network Configuration Protocol (NETCONF). RFC 6241 (Proposed Standard), June 2011.
[8] 情報通信研究機構. JGN-X パートナーシップ・サービス：IP 仮想化サービス. https://www.jgn.nict.go.jp/nwgn/ja/reports/documents/03_IP_virtualized-service.pdf (参照 2016-01-30).