IPネットワークにおける特定経路の可視化

小 原 泰 弘[†1]

本論文は，インターネットの信頼性向上のため，ネットワークにおける経路制御システムについて，経路の確認，監視を容易にする手法に取り組む．具体的には，ネットワーク内で動作中のルータからSNMPを利用してOSPF LSDBを取得し，独自にDijkstra計算を実行してネットワーク全体の経路を計算し，ある特定の宛先への経路をGraphVizで図示するという手法を検討する．本手法の特徴はトポロジや経路を単純にグラフ構造として抽象化しAPIとして利用するという設計である．本手法は，研究のみならず，ネットワークの管理・運用に活用されることが期待できる．プロトタイプ実装，StarROUTEを実装し，実際のネットワークにおける経路を図として示した．現状ではネットワークの管理・運用に貢献することは未だ不可能であるが，その拡張性に期待できる．

Visualization of Specific Routes in IP Network

YASUHIRO OHARA[†1]

This paper proposes a method that retrieves OSPF LSDB via SNMP from a running router, calculates routes using Dijkstra computation, and visualize routes to a destination by GraphViz. With the design principle such that network topologies and routes be abstracted as simple graph structures and used as the API, we implemented a prototype called "StarROUTE". This is expected to be utilized not only in research but also in network administration and operation. Although it is not feasible to contribute to the network administration and operation as is, the extensibility is hopeful.

1. はじめに

近年では，インターネットはどこでも高速に接続できるのが前提となってきた．ネットワーク回線の速度は，これまでに，1996年 6Mbps，2000年 2.4Gbps[23]，2010年 100Gbps[4] と劇的に高速化している．また，IPデータネットワークとしてのインターネットの普及率は，2009年時点で，全世界では1/4 (25.9%[8])，国内の世帯で60%以上[25]に達している．これらはネットワークの高速化と広域化の一部の証左でしかないが，携帯電話の普及などにもより，インターネットサービスやそれを実行するソフトウェアは「高速なネットワーク接続がどこでも使える」ことを前提するようになってきた．その一例にTwitter[20]が挙げられる．

また，クラウドコンピューティングと呼ばれる，遠隔データセンター処理型のサービスが増えつつある．グーグルのインターネット検索，地図アプリケーション，スパムフィルタを利用した不特定多数へのメールサービスや，アマゾンの書籍検索，レコメンデーションサービスなどでは，大量のデータを処理する必要がある．近年のCPUの小型化，並列化による消費電力や発熱量の増加から，これらの業者では電源や冷却の設備を集中させ効率化する必要が出てきた．加えて，ユーザ側からのこれらサービスへの要求が，質・量・種類ともに増えてきたことにより，個人ユーザの環境では実現不可能な計算処理をデータセンターで代替し，結果のみを個人環境へ返答することが求められている．これがクラウドコンピューティングと呼ばれる形態である．

アプリケーションが高速・広域なインターネット接続を前提とし，多くのサービスがネットワーク越しに計算を実行するクラウドコンピューティングへ移行している今日のインターネットサービスにおいて，データを中継するネットワーク部分への依存度は日増しに増加していると言える．つまり，個人環境とデータセンターを接続するインターネットサービスプロバイダ (ISP) のネットワークの高信頼性が今まで以上に求められる．また，データセンター内部のネットワークの信頼性も新しく重要となってきた．

ネットワークの信頼性は，経路制御システムが計算する経路の信頼性であるとも言える．何故ならば，パケットロス率や遅延の大きいネットワーク部分は，経路制御システムが賢く動作していれば (原理的には) 回避できるはずだからである．つまり，経路が意図通りに設定 (計算) された状態であるかどうかを確認する

†1 北陸先端科学技術大学院大学 情報科学センター
Center for Information Science,
Japan Advanced Institute of Science and Technology

ことは，ネットワークの信頼性に直接関わる重要な作業である．本論文では，経路制御システムが計算する経路を確認・検証することに焦点を当てる．

現状の経路の確認手法は，煩雑であり，そのため規模性に欠ける．このことは，経路の確認の迅速さに影響する．そして，故障箇所を特定し，それを修正する，もしくは経路変更により回避するという，通信可能性の回復作業を遅め，結果的にネットワークの信頼性を低下させる．これまで，ネットワーク全体で特定の経路がどのような状態になっているかを確認することは，主に以下の 2 つの手法で行われてきた．

(1) ssh などでルータの制御端末にアクセスし，各ルータでそれぞれ，そのルータでの対象の経路がどうなっているか調べる．この手法は，各ホップのルータにアクセスし経路を調べなければいけないため，規模性に欠ける．

(2) traceroute コマンドを利用して，送信元ホストから宛先への経路上のルータのリストを表示する．この手法では，ICMP 応答を返さないルータが検知できない，通信の往路しか検知できない，故障箇所から向こう側の経路を知ることができない，これらの原因から故障箇所を特定することができないなど，様々な問題点がある．また，経路のフラップなどをこの手法で検知することは困難である．

経路に限らないネットワークの状態確認のためには，Network Management System (NMS) が広く利用されている．これには IBM Tivoli[7]，HP OpenView[6]，CiscoWorks[2]，OpenNMS などがある．機能としては，SNMP でネットワーク内の装置にアクセスし，情報を取得し，ウェブページとして表示するのが典型的であり，ネットワーク管理に利用される．これらは，経路の確認という特化された目的には不十分である．

本論文では，ネットワーク内で動作中のルータから SNMP[5] を利用して OSPF[11] LSDB を取得し，独自に Dijkstra 計算を実行してネットワーク全体の経路を計算し，ある特定の宛先への経路を図示するという手法をとった．類似の方法には，Zebra ospf6d を利用したトポロジ情報の収集手法[28](以下「OSPFv3 を利用したトポロジ収集」と呼ぶ．)，クラウドスコープテクノロジー社の PATHMANAGER OSPF モジュール[24] がある．これらとの違いは，4 章で考察する．

本手法の利点と欠点を概観する．利点としては，以下が挙げられる．

(1) 高速，高効率である．一台のルータにアクセスし，ネットワーク全体の経路を独自に計算できるため，
  (a) ネットワークへの負荷が小さい．
  (b) ルータ数に関するネットワーク規模の拡大に対応できる (各ホップの制御端末にアクセスする上記 (1) の手法に比べ)．
  (c) Dijkstra の計算が高速であるため，高速に処理できる．

(2) OSPF LSDB を取得しているため，OSPF コストなどより詳細なパラメータを確認できる．

(3) 図示することにより，ネットワーク管理者が把握することが容易である．

(4) 高速性から，経路のフラップなど時系列の変化に対応できる可能性がある．これをするための方法には，複数の図を (パラパラ漫画のように) アニメーションして検知・確認する手法が利用できる．

(5) 終点を変更した複数の経路にも個別に対応できる．

欠点としては，以下が挙げられる．

(1) 独自に経路を計算しているため，本当に利用される・設定された，実際の経路とは異なる可能性がある．

(2) リンクステート型 (分散データベース型) 経路制御プロトコルの特徴を利用しているため，距離ベクタ型 (分散計算型) とその派生には応用できない．つまり，OSPF，IS-IS[1] には応用できるが，RIP[9]，BGP[16]，(E)IGRP には応用できない．

(3) リンクステート型経路制御プロトコルが持つデータベースに依存する．実際のネットワークには存在するが (例えば)OSPF が動作していないようなルータは無視される．また，リンクステート型経路制御プロトコルは AS 内で動作するのが一般的であるため，AS を越える経路は確認できない．

本手法を実装し StarROUTE[22] と名付けた．以下，2 節で実装を解説し，3 節で実行結果の例の図を示す．4 節で考察，5 で将来の展望を述べる．6 で本論文をまとめる．

2. 実　　装

2.1 概　　要

StarROUTE の名は，StarBED[26] から名付けた．StarBED は，NICT 北陸リサーチセンター[27] に構築されたシステム検証のためのテストベッドである．920 台の PC が設置されており，構成，管理，運用はデータセンターに類似する*1．StarROUTE 開発の背景・動機には，StarBED における来るべき経路制御システムの検証の際に，確認・評価のために規模性を持つ経路検証用ツールが必要であろうと想定したことがある．さらに，この経路の検証確認という行為の対象および目的を拡大すれば，StarROUTE の手法は一

*1 本論文では，システム検証という特化した目的を持つため，一般的なデータセンターにはテストベッド施設を含まないと定義した．

表 1 GraphViz オプション．アスタリスク (*) は default を示す．
Table 1 GraphViz option. An asterisk(*) indicates the default.

レイアウト	dot	fdp	sfdp*	circo	twopi
出力フォーマット	dot	png	eps	all*	

般的なネットワークの管理運用にも活用できるのではないかと思われる．これが本論文の主題である．

StarROUTE は，以下の機能を持つ．

(1) Net-SNMP[12] を利用してルータの OSPF LSDB を取得する．
(2) OSPF LSDB からネットワークトポロジをグラフ構造として表現する．
(3) 指定された終点に対して，Dijkstra により経路を計算する．この経路もグラフ構造として表現した．
(4) グラフ構造のノード位置を GraphViz[3] により計算させる (レイアウト)．
(5) グラフ構造を GraphViz により図として出力する．

上記機能のうち 4 および 5 で選択可能な GraphViz のレイアウトアルゴリズムと出力フォーマットを**表 1** に示す．出力フォーマット “all” のみ StarROUTE 独自のもので，他の全ての出力フォーマットを個別に，複数の図を出力する．

StarROUTE のプロトタイプ実装は，参照 22) から取得可能である．

2.2 利用方法

StarROUTE のコマンドラインオプションを以下に示す．

StarROUTE 実行オプション

```
./starroute -h <SNMP エージェントホスト> -c <SNMPv2c コミュニティ名> -d <終点ホスト指定> -l <レイアウトアルゴリズム> -o <ファイル名のプリフィックス> [-r] [-n] [-w]
```

`[-r]`，`[-n]` はそれぞれ，ルータ，ネットワークに説明ラベルを表示する．`[-w]` は，リンクやルータに OSPF コストを表示する．

実行例を以下に示す．

StarROUTE 実行例

```
./starroute -h 150.65.0.1 -c community -d 150.65.0.1
-l sfdp -o JAIST-starroute-sfdp-rnw- -r -n -w
```

この例では，150.65.0.1 に SNMP アクセスし，OSPF バックボーンエリアの LSDB を取得し，150.65.0.1 を終点とした経路を計算し，sfdp レイアウトで dot, png，eps ファイルを出力する．ネットワークには説明ラベルが表示され，リンクには OSPF コストが，ルータには “説明ラベル<終点までのコスト>” が表示される．このコマンドの実行結果は後述する**図 3** となる．結果のファイル名は，JAIST-starroute-sfdp-rnw-topology.{dot,eps,png}，JAIST-starroute-sfdp-rnw-spf-to-150.65.0.1.{dot,eps,png} となる．

```
Usage: starroute [OPTION...]

-v, --snmp-version     Set SNMP version. Only SNMPv2c
                       is supported. default: 2c.
-h, --snmp-target      Set SNMP target.
                       default: 127.0.0.1.
-c, --snmp-community   Set SNMP community.
                       default: "ro".
-a, --ospf-area        Set SNMP area
                       default: 0.0.0.0.
-l, --layout           Specify Graphviz layout: {dot,
                       fdp,sfdp,neato,...}
                       default: "sfdp".
-o, --output-prefix    Set output file prefix.
                       default: "out/starroute-outout-".
-t, --output-type      Specify Graphviz output format:
                       {dot,eps,png,...,all}
                       default: all.
-d, --destination      Set destination key
-r, --router-label     Enable displaying router-label
-n, --network-label    Enable displaying network-label
-w, --ospf-weight      Enable displaying ospf-weight
-V, --version          Print program version
-H, --help             Display this help message

Report bugs to yasu@jaist.ac.jp
```

図 1 StarROUTE コマンドラインヘルプ．
Fig. 1 StarROUTE command-line help.

表 2 StarROUTE の動作環境．
Table 2 Configuration of StarROUTE.

名前	バージョン
Mac OS X	10.6.4 (10F569)
NET-SNMP	5.5
GraphViz	2.26.3 (20100126.1600)

コマンドラインヘルプを**図 1** に示す．

StarROUTE(バージョン 0.0.1c) の動作環境を**表 2** に示す．

2.3 libGVE ライブラリ構想

トポロジや経路を単純にグラフ構造として抽象化できれば，入力プロセスと出力プロセスへの柔軟なインターフェイス (API) となる．StarROUTE はこの設計のもと実装した．入力プロセスには，執筆時点で，OSPF LSDB からグラフ構造 (トポロジ) を作成する ospf モジュールと，グラフ構造 (トポロジ) からグラフ構造 (経路) を作成する dijkstra モジュールがある．出力プロセスには，グラフ構造 (トポロジおよび経路) から図を出力する graphviz モジュールがある．

後述 (5 節参照) するように，将来さらに拡張モジュールを実装するため，グラフ構造関連のモジュール (主に `graph.[ch]` および `attr.[ch]`) は，後に libGVE

という名前で切り出す予定である．libGVE という名前は，単純に $G = (V, E)$ のグラフ構造としてアクセスできるようにするという設計思想から名付けた．

libGVE は，以下のようにグラフにアクセスすることができる．

グラフ G の頂点集合 V の反復

```
struct set_node *n;
for (n = set_head (G->V); n; n = set_next (n))
  {
    struct vertex *v = (struct vertex *) n->data;
    v に対する操作;
        :
  }
```

頂点 t, v, w や辺 e へのアクセス

```
struct set_node *n;
struct vertex *v, *w;
for (n = set_head (v->outgoing); n; n = set_next (n))
  {
    struct edge *e = (struct edge *) n->data;
    assert (e->source == v);

    w = e->sink;
    v,w に対する操作;
        :
  }
```

attr モジュールを利用して，グラフ構造，頂点，辺には，任意の個数の Type-Length-Value(TLV) 属性を設定することができる．執筆時点で，型には `int` 型，文字列 (`char *`)，IPv4 アドレスに対応している．Dijkstra アルゴリズムのラベルなどはこれを利用して実装した．

グラフ G，頂点 t, v，辺 e に対する属性操作

```
struct graph *G;
struct vertex *t, *v;
struct edge *e;
int c, weight, vlabel;

attr_set_printf (G->attrs, "graph-name", "topology")
attr_set_string (t->attrs,
                 "dijkstra-type", "destination");
attr_set_int (e->attrs, "ospf-weight", weight);
attr_set_int (v->attrs, "dijkstra-label", vlabel++);
c = attr_get_int (v->attrs, "dijkstra-weight");

if (attr_get_int (e->source->attrs,
                  "dijkstra-label") > 0)
  {
     :
  }
```

3. 実 行 例

StarROUTE の JAIST での実行結果の図を例とし

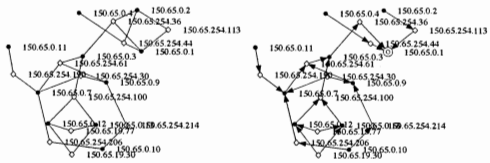

図 5 JAIST におけるトポロジ (左)，ある終点への経路 (右) の図．(fdp レイアウト)

Fig. 5 Topology(left), and routes to a router(right) in JAIST. (layout: fdp)

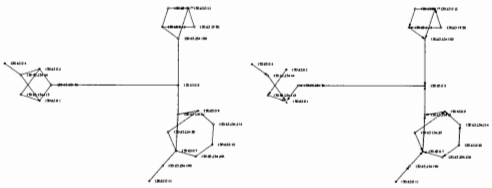

図 6 JAIST におけるトポロジ (左)，ある終点への経路 (右) の図．(circo レイアウト)

Fig. 6 Topology(left), and routes to a router(right) in JAIST. (layout: circo)

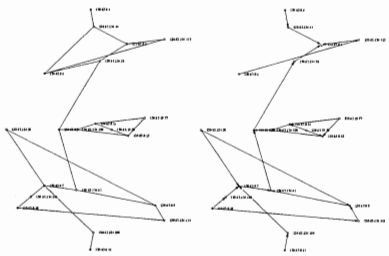

図 7 JAIST におけるトポロジ (左)，ある終点への経路 (右) の図．(twopi レイアウト)

Fig. 7 Topology(left), and routes to a router(right) in JAIST. (layout: twopi)

図 2 JAIST におけるトポロジ (左)，ある終点への経路 (中央)，他の終点への経路 (右) の図．(sfdp レイアウト)
Fig. 2 Topology(left), routes to a router(middle), and routes to another router(right) in JAIST. (layout: sfdp)

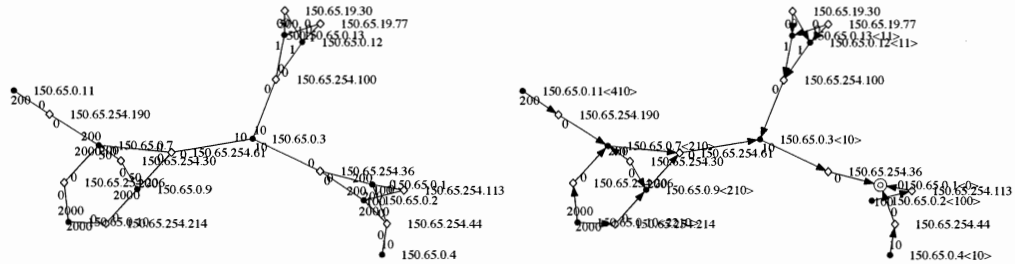

図 3 JAIST におけるトポロジ (左)，ある終点への経路 (右) の図．(sfdp レイアウト，説明ラベル表示)
Fig. 3 Topology(left), routes to a router(middle), and routes to another router(right) in JAIST. (layout: sfdp, with description label)

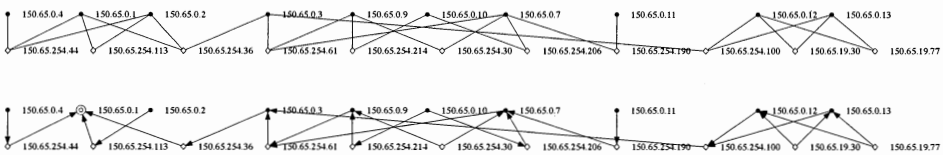

図 4 JAIST におけるトポロジ (上)，ある終点への経路 (下) の図．(dot レイアウト)
Fig. 4 Topology(top), and routes to a router(bottom) in JAIST. (layout: dot)

て示す．**図 2**，図 3，**図 4**，**図 5**，**図 6**，**図 7** は，すべて JAIST のネットワークトポロジか，その上で計算した経路を示しており，図 2 の右図を除いて，すべて同じグラフ構造を示している．異なるのは，GraphViz のレイアウトアルゴリズムのみである．

図 2 は，sfdp によるレイアウトを利用して，トポロジ図と，異なる二つの終点への経路を示している．sfdp は，もっとも見やすいアルゴリズムに思える．

図 2 の左の二つの図 (つまり，トポロジ図と経路) について，ルータ，ネットワーク，OSPF コストの 3 つの説明ラベルを全て表示させたものが，図 3 である．自動レイアウトを利用している場合には，どれがどのルータであるのか，説明ラベルを表示しないと理解できない．この例では，ラベルをつけたことにより，図が煩雑になり，ところどころ重なりあって解読できない結果となった．

その他のレイアウト，dot，fdp，circo，twopi の図を，それぞれ図 4，図 5，図 6，図 7 に示す．ネットワークの管理運用に利用するという観点からは，少なくともルータ，ネットワークの識別は必須であるため，それらの説明ラベルは全てに付与した．その他のレイアウトアルゴリズムでは，余白や作図のバランスが悪く，グラフ構造を理解するのにさらに非効率的であるように思われる．

図 8 に，WIDE Project[21] のネットワークを図示したものを示す．ここでは，説明ラベルを表示すると，グラフ構造が理解できなくなるほど煩雑になる．説明ラベルは省いたが，そのため，ノードの識別ができず，経路がどのように向いているのか理解できない．これでは結局目的が果たせているとは言えない．

4. 考　　察

GraphViz の自動レイアウトは，現状では効果的でなく，グラフの構造と経路の状態がどのようになっているのかを同時に読み取ることは困難であった．ここ

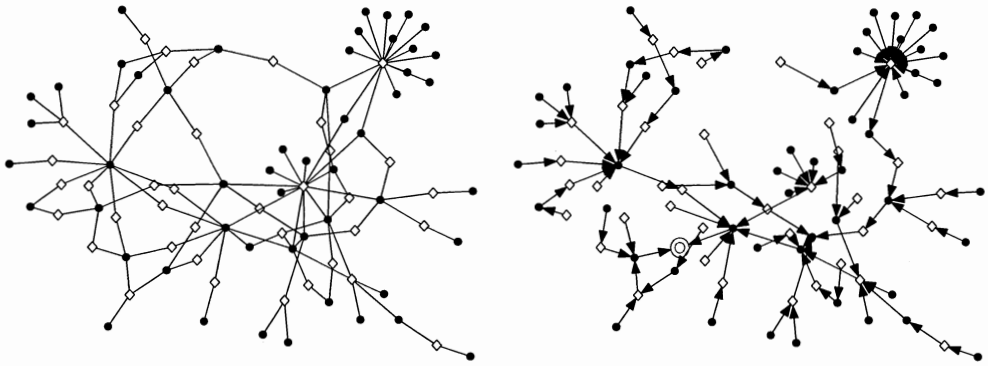

図 8 WIDE Project におけるトポロジ (左)，ある終点への経路 (右) の図．(sfdp レイアウト)
Fig. 8 Topology(left), and routes to a router(right) in WIDE Project. (layout: sfdp)

で，レイアウトを手動で行えば，GraphViz を使う意味は半減する．StarROUTE は，現状のままでは経路の確認ができず，検証や運用管理に利用することはできないことが分かった．

一方で，本手法は効果的であることも分かった．今回の図は，一度のコマンド実行で 2 枚ずつ出力できる．この時間的，労力的な負担の軽減は，この手法，正確にはリンクステート・分散データベースの特徴を活用する方法は，とても効果的である．

libGVE の抽象的グラフ構造を利用した API は，さまざまな自動レイアウト技術に容易に対応することを実現可能にする．TomSawyer[18] などの他の自動レイアウトソフトウェアに対応すれば，この問題は克服できると思われる．

前述した，OSPFv3 を利用したトポロジ収集[28]，クラウドスコープテクノロジー社の PATHMANAGER OSPF モジュール[24] との違いを以下に述べる．OSPFv3 を利用したトポロジ収集は，異なる OSPF エリアのトポロジ収集を目的とした．PATHMANAGER OSPF モジュールは，運用に活かすために MPLS や VPN との協調を目的としている．本論文はこれらをネットワーク運用に活かすためにもう一歩進め，トポロジ図を様々なレイアウトで表示する多様性や，一度のコマンド実行で図を複数枚保存する効率性などの利便性を重視した手法であるということができる．libGVE を利用した抽象的グラフ構造による API は，新しいグラフ表示ソフトウェアに対応することを容易にする．これは，トポロジ図を自動表示した際の可視性を高める他の研究成果を効果的に利用できるという点で，好ましい特徴である．

5. 拡張性と展望

libGVE の抽象的グラフ構造を利用した API は，グラフアルゴリズムに関する研究を促進するという目的では，とても有効である．著者はこれまでの研究活動[13)-15)] において，本提案と同様なカスタムシミュレータソフトウェアを開発したが，その時の経験と反省から再開発したものが StarROUTE/libGVE である．これを利用すれば，BRITE[10] のようなグラフ生成ソフトウェア，RocketFuel[17] のようなグラフ構造データに対応すること，さらに，グラフ生成アルゴリズムを自作することも容易になる．他にも，I-VT[19] などの既存の信頼性計算アルゴリズム，MARA[14] のような新しい経路制御アルゴリズムを開発することが可能になる．libGVE による抽象的グラフ構造によるソフトウェアモジュールの分離は，グラフに対する入力モジュール (SNMP,BRITE,RocketFuel) と，グラフに対する計算モジュール (Dijkstra,MARA,I-VT)，およびグラフに対する出力モジュール (GraphViz) を別々に開発し，統合することを容易にする．将来的には，グラフ構造を解析する新しいアルゴリズムを開発してネットワーク設計などの運用に役立てることも期待できる．

6. ま と め

本論文では，抽象的グラフ構造を API として利用し，OSPF LSDB のグラフ構造を GraphViz のグラフ構造に変換する手法を検討した．これは，研究のみならず，ネットワークの管理・運用に活用することが期待できる．現状ではネットワークの管理・運用に貢献することは未だ不可能であるが，その拡張性に期待できる．

参 考 文 献

1) : Intermediate system to Intermediate system routeing information exchange protocol for use in conjunction with the Protocol for providing the Connectionless-mode Network Service (ISO 8473), ISO/IEC 10589:2001, ISO (2001).

2) Cisco Systems, Inc.: Ciscoworks - Cisco - Cisco Systems, http://www.cisco.com/en/US/products/sw/cscowork/ps2425/index.html⟩.
3) Ellson, J. and Gansner, E.: Graphviz, http://www.graphviz.org/.
4) Geek なページ：世界初の 100Gbps 1 波伝送実運用@ Interop, http://www.geekpage.jp/blog/?id=2010/6/11/1 (2010).
5) Harrington, D., Presuhn, R. and Wijnen, B.: An Architecture for Describing Simple Network Management Protocol (SNMP) Management Frameworks, RFC 3411 (2002).
6) Hewlett-Packard Development Company, L.P.: HP Network Node Manager (NNM) Advanced Edition software, https://h10078.www1.hp.com/cda/hpms/display/main/hpms_content.jsp?zn=bto&cp=1-11-15-119^1155_4000_306__.
7) IBM: IBM Tivoli Software, http://www-306.ibm.com/software/tivoli/⟩.
8) ITU: World Telecommunication/ICT Indicators database, http://www.itu.int/ITU-D/ict/statistics/.
9) Malkin, G.: RIP Version 2, RFC 2453 (1998).
10) Medina, A., Lakhina, A., Matta, I. and Byers, J.: BRITE: an approach to universal topology generation, *MASCOTS*, Vol.00, p.0346 (2001).
11) Moy, J.: OSPF Version 2, RFC 2328, IETF (1998).
12) Net-SNMP: Net-SNMP, http://net-snmp.sourceforge.net/.
13) Ohara, Y.: Routing Architecture for the Dependable Internet, Ph.D. dissertation, Keio University (2008).
14) Ohara, Y., Imahori, S. and Meter, R. V.: MARA: Maximum Alternative Routing Algorithm., *INFOCOM*, IEEE, pp.298–306 (2009).
15) Ohara, Y., Kusumoto, H., Nakamura, O. and Murai, J.: Drouting Architecture: Improvement of Failure Avoidance Capability Using Multipath Routing, *IEICE Transactions*, Vol.91-B, No.5, pp.1403–1415 (2008).
16) Rekhter, Y., Li, T. and Hares, S.: A border gateway protocol 4 (BGP-4), RFC 4271, IETF (2006).
17) Spring, N., Mahajan, R., Wetherall, D. and Anderson, T.: Measuring ISP topologies with Rocketfuel, *IEEE/ACM Trans. Netw.*, Vol.12, No.1, pp.2–16 (2004).
18) Tom Sawyer Software: Data Visualization Software — Tom Sawyer Software, http://www.tomsawyer.com/home/index.php.
19) Tong, L. and Trivedi, K.S.: An improved algorithm for coherent-system reliability, *IEEE Transaction on Reliability*, Vol. 47, No. 1, pp. 73–78 (1998).
20) Toru Saito: 【ニールセン調査】日本の最新ソーシャルメディア・トレンド - ブログとツイッター普及は世界トップ，Facebook も 3 %超, http://blogs.itmedia.co.jp/saito/2010/07/tweet.html.
21) WIDE Project: WIDE PROJECT, http://www.wide.ad.jp/.
22) Yasuhiro Ohara: StarROUTE tarball, http://www.jaist.ac.jp/~yasu/starroute-0.0.1c.tgz.
23) インプレス R&D：NETWORLD+INTEROP で高速ネットワークが体験できる，*INTERNET magazine* (2000).
24) 株式会社クラウド・スコープ・テクノロジーズ：PATHMANAGER OSPF モジュールをリリース　日本初，OSPF ルーティングプロトコルを監視，http://www.cloud-scope.com/news/pdf/090525CST.pdf (2009).
25) 財団法人インターネット協会：インターネット白書 2009，インプレス R&D (2009).
26) 情報通信研究機構 北陸リサーチセンター：StarBED, http://www.starbed.org/.
27) 情報通信研究機構北陸リサーチセンター：北陸リサーチセンター，http://starbed.nict.go.jp/.
28) 有田敏充，飯田隆義，吉田和幸：IPv6 ネットワークにおける複数の OSPF エリアからのトポロジ情報収集について，情報処理学会研究報告，Vol.2009-IOT-5, No.13, pp.1–6 (2009).