ネットワークエミュレーションテストベッドを用いた 実 OSPF トポロジ模倣システム

鈴木 未央† 櫨山 寛章‡ 榎本 真俊‡ 三輪 信介† 門林 雄基†‡

† 情報通信研究機構
‡ 奈良先端科学技術大学院大学

要 旨

インターネットに新しい技術を導入するためには、その技術が妥当であるのか、悪影響を及ぼさないのかなどを、実践的に検証する必要がある。特に広くインターネット上に展開する予定の技術について検証するためには、実際のインターネットに近い現実的な環境が求められる。そこで、このような技術の実験を支援するために、我々はテストベッド上に模倣インターネットを構築する試みに取り組んでいる。本稿では、テストベッド上に AS 内 OSPF ネットワークを模倣し構築するための手法と試行について述べ、その考察を行う。試行では、6 ルータの小規模な OSPF ネットワークについて実用的な時間で模倣できることを確認した。

A Topology Imitation System from A Real OSPF topology to A Network Emulation Testbed

Mio SUZUKI† Hiroaki HAZEYAMA‡ Masatoshi ENOMOTO‡
Shinsuke MIWA† Youki KADOBAYASHI†‡

†National Institute of Information and Communications Technology
‡Graduate School of Information Science, Nara Institute of Science and Technology

Abstract

In order to evaluate new technologies which will be introduced to the Internet, these technologies should be practically experimented to confirm their adequacy and whether or not there are sideeffects. To experiment some of these technologies, which will be widely deployed on the Internet, realistic environment which seems similar to the Internet are demanded. In order to support to experiment on realistic Internet like environment, we are now trying to make the Internet on a testbed. In this paper, we describe our method to imitate a real-operated OSPF network on the testbed, and we also report our trials to imitate a OSPF network on a small testbed and StarBED with our tools. We confirmed that a small OSPF network was imitated with 6 testbed nodes in a short time.

1 はじめに

インターネットのような大規模な分散環境に新しい技術を導入する場合、その技術がインターネット上で適切に動作するのか、既存のインターネット環境に悪影響を及ぼさないのかなどを、バイナリ実装レベルで実践的に検証しておく必要がある。このような検証を行うためには、現実のインターネットに則した実験環境が必要である。このため、ネットワーク実証実験用クラスタ (ネットワークエミュレーションテストベッド、以下 NET) が多くの研究者や開発者の実験に利用されている。NET とはネットワーク実証実験を目的とした PC クラスタであり、クラスタ内のレイヤ 2 スイッチを設定してノードをルータとして動作させることで実験者の意図するネットワークトポロジを構築できる。実験者は構築されたネットワークトポロジの上のノードでバイナリ実装を動作させ実験を行える。日本国内の代表的な大規模 NET としては 1000 台以上のノードを備える StarBED [1][2] がある。

これまで我々は AnyBed [3] や XENebula [4] の研究を通して NET における AS レベルの BGP ネットワーク実験環境構築の省力化に取り組んできた。AnyBed は、RouteViews Project [5] や CAIDA Project [6, 7] などで公開されている AS 接続関係データセットを元にして、実運用されている AS レベルネットワークを模倣した実験用 BGP ネットワークの構築とそのネットワーク上での実験を省力化するためのツールセットである。本稿での模倣とは、元のネットワークトポロジが持つ構造、AS 番号、IP アドレス情報などをなるべくそのままの形で実験環境上に再現することである。 AnyBed は StarBED のような 1000 台規模の大規模 NET から研究室レベルの数台規模のクラスタまで対応し、使用ノード数が 100 台程度のネットワークであれば数分で実験ネットワークトポロジを構築できる [3]。

これまで AnyBed では AS レベルの BGP ネットワークを模倣し構築することに注力してきたが、実験者からは AS レベルネットワークだけではなく AS 内ネットワークも模倣してほしいとの要望が寄せられていた。このため、我々は AnyBed の構築機能を拡張し、AS 内で実運用されている OSPF ネットワークを模倣する実験ネットワークの構築機能を設計し実装した。その際の目的として、ネットワーク上で動作するバイナリ実装の評価のみならず、ネットワークの障害時の挙動調査やネットワーク設計者・オペレータの教育にも利用できるような AS 内 OSPF ネットワークの正確な模倣を目指した。

本稿では、この実 OSPF ネットワークを NET 上に模倣するための実 OSPF トポロジ模倣システムを提案し、その実装による実験ネットワークの構築の試行と考察を行う。

2 関連研究

実験者が OSPF ネットワークを構築し利用できるテストベッドとして DETER [8] や Remote Network Labs [9] がある。また、OSPF のシミュレーション機能を持つソフトウェアシミュレータとして ns-2 [10]、SSFnet [11]、SimRouting [12] がある。

OSPF トポロジに限らず、特定のモデルによりインターネットのトポロジを生成できるソフトウェアとして、Inet [13]、GT-ITM [14]、BRITE [15] などのトポロジジェネレータがある。更に観測結果をもとに推測される AS 内を含む実トポロジを公開している Rocketfuel [16] のようなプロジェクトがある。

これらのテストベッドやソフトウェアのうち DETER の設備と Rocketfuel の提供するデータセットを用い、OSPF や BGP トポロジの構築行うためのソフトウェアとして Rocketfuel-to-ns [17] がある。しかし、Rocketfuel で採用されている手法は traceroute の結果をもとにトポロジを推測しており、OSPF コストやリンク情報の正確な収集には限界があると考えられる。本研究が目的としているネットワーク障害時の挙動調査やネットワーク設計者・オペレータの教育のためには、実トポロジの OSPF コストやリンク情報などが正確に反映された模倣ネットワークが必要であるため、これらの情報をより正確に収集できる手法が求められる。

他にも、ネットワーク運用支援のために OSPF トポロジを可視化するソフトウェアや研究として ospfviz [18] やネットワーク構成情報表示システムのための自動配置アルゴリズムの評価 [19] がある。

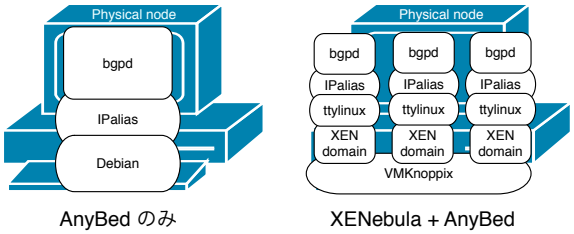

図 1: AnyBed の動作環境

3 実 OSPF トポロジ模倣システムの設計と実装

実 OSPF トポロジ模倣システムの設計について述べるにあたり、まず我々が過去に研究を行ってきた実験トポロジ構築システムである AnyBed の概要を述べる。その後、AnyBed の拡張である本模倣システムの設計と実装について述べる。

3.1 AnyBed 全体の概要

AnyBed が動作するために必要とされる環境を図 1 に示す。AnyBed ではクラスタノード上で Debian もしくは ttylinux を動作させ、IP alias と Quagga [20] に含まれる bgpd により実験用の BGP ネットワークを構築する。また、AnyBed と XENebula を併用することにより、VMKnoppix と XEN による OS 仮想化を用いて実験に利用可能なノード数を増やす事ができる。また、実験に利用するノードの OS は PXE boot と NFS により AnyBed や XENebula のサーバから提供されるため、ノードの HDD にインストールされている OS には依存しない。このため、NET 用のクラスタだけではなく他目的のクラスタを一時的に借りて実験を行う事もできる。

AnyBed 全体の設計を図 2 に示す。AnyBed は三つの機能から構成される。一つめは実験ネットワーク構築のための情報を集める部分であるデータ収集部、二つめは NET に存在するノードなどの資源割り当てを行う資源割り当て部、三つ目は設定ファイルを NET のノードに配布し、実際の実験ネットワークを構築する設定反映部である。各部を構成するコンポーネントは容易に交換可能である。データ収集部と資源割り当て部のデータ交換には XML で記述されたファイルを用いる。

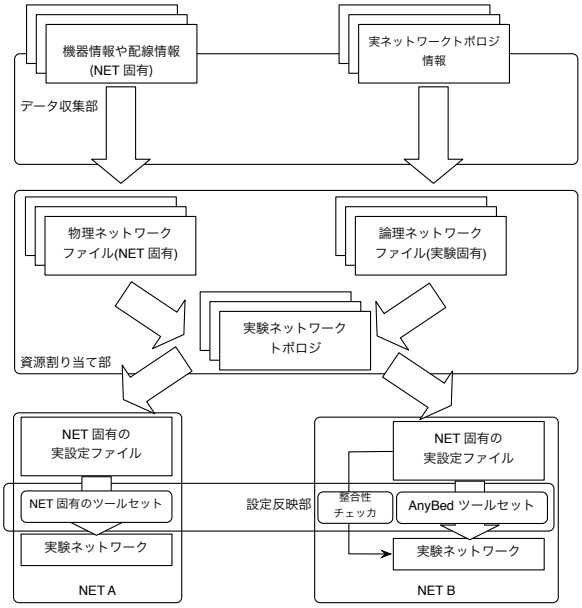

図 2: AnyBed の概念図

AnyBed では実験ネットワークのトポロジに関する情報を論理ネットワークファイルと物理ネットワークファイルに分離する。論理ネットワークファイルは実験ネットワークのレイヤ 3 トポロジを記述する。このため、論理トポロジは特定の NET には依存しない。この論理ネットワークファイルの生成方法として、実験者が手で作成する他に、CAIDA Project で公開されているデータセット を AnyBed に含まれるフィルタツールによって加工し生成することもできる。一方、物理ネットワークファイルには NET 固有の情報である物理ノード、配線、ネットワークインタフェイスの帯域などの情報を記述する。この物理ネットワークファイルは NET 毎に存在する機器構成表や管理台帳などから、AnyBed に入力できる形式に変換することを想定している。

実験を行う際には、まず物理ネットワークファイルを元に論理ネットワークファイルに対してノードなどの資源割り当てを行う。この割り当てにおいて、可能な限り各リンクの帯域が公平となるようにノードとリンクが割り当てられる (文献 [3] 5.2 節)。この割り当ての結果、各ノードの OS 固有の設定ファイルとルーティングデーモンの設定ファイルなどが生成される。その後、生成された設定ファイルを NET の各ノードとレイヤ 2 スイッチに配布し、各ノードにて IP alias による IP アドレス付与とルーティングデーモンの起動などを行い、実験ネットワークが

構築される。最後に、実験ネットワークが正しく構成されているかどうかを設定ファイルと照らし合わせてチェックする。

3.2 実 OSPF トポロジ模倣システムの設計

我々が実 OSPF トポロジ模倣システムを設計するにあたり、要件として考えたのは以下の通りである。

要件 1. AS 内や組織内の一般的な OSPF ネットワークを IP アドレスや OSPF コストなどの情報を正確に模倣した状態で NET 上に構築できること

要件 2. 実用的な時間で模倣した実験ネットワークが構築できること

要件 3. 既存の AnyBed のツール類や環境の拡張とするため、これらと親和性が高い設計とすること

要件 1 については、一般的な OSPF ネットワークというものは定義するのが難しいが、おおむね数台から数十台程度のルータで構成されたネットワークを模倣することを想定する。これは、我々が所属し本研究の検証に利用しようと考えている WIDE プロジェクトの OSPF ネットワークが 50 個程度のルータで構成されており、当面はこの程度の規模のネットワークの模倣を目標としていることによる。また、これまで AnyBed で BGP ネットワークを構築する際には、マクロ的な AS の接続関係の情報を重視し、IP アドレスについてはプライベート IP アドレスを利用し実験ネットワークを構築していた。しかし、OSPF ネットワークの模倣にあたっては BGP ネットワークの模倣よりも模倣粒度の細かさが必要であり、また本研究の目的達成のためには詳細な個々のネットワークの IP アドレス情報、リンク情報、個々のルータの情報が重要になると考えられることから、実ネットワークの情報を実アドレスも含めた形で模倣することを目標とする。

要件 2 については、クラスタノードの電源が入っており OS が起動している状態を前提として、トポロジ収集、設定生成、反映の後、実験ネットワークが構築されて実験可能な状態になるまでに要する時間を数十分以内にすることを目標とする。これは従来の AnyBed からの目標を引き継いだものである。

要件 3 について、図 2 で示される AnyBed の拡張点として、データ収集部に OSPF ネットワークからトポロジに関する情報を収集する機能を追加すること、資源割り当て部に ospfd の設定を生成する機能を追加すること、の大きく二点がある。この二点について順に述べていく。

本研究の目的は実運用されている OSPF ネットワークを模倣して NET 上に構築することであるため、実運用されている OSPF ネットワークからネットワークトポロジに関する情報を収集する手法が重要である。この収集手法の候補として我々が考えたものは以下の通りである

手法 1. 観測ポイントから別のポイントに traceroute を行い、その結果からトポロジを推測する

手法 2. OSPF ネットワークを構成しているすべてのルータから設定と Routing Information Base(以下 RIB) と Forwarding Information Base(以下 FIB) を取得して解釈を行いトポロジを推測する

手法 3. 同様にある 1 台のルータから OSPF リンクステートデータベース (以下 OSPF LSDB) の内容を取得し、解釈を行いトポロジを推測する

本研究では手法 3 を採用する。その理由を以下に述べる。手法 1 は Rocketfuel と同等の方式であるが、経路のベストパス情報しか得られずコストを含めたトポロジが収集できないため、トポロジの正確な模倣を目指す本研究では採用を見送る。また手法 2 は、OSPF ネットワークが複数ベンダの機器で構成されている場合、多数の機器から情報収集を行わねばならず、収集を行う機器毎に必要な情報を取得するための SNMP・NETCONF のサポート状況やログイン後のデータ取得コマンドが異なり、汎用的な情報取得を行うことが難しいと考えられるため採用を見送る。

手法 3 を採用するにあたり、ルータから OSPF LSDB を取得する手法として、ルータにログインしてコマンドを発行しその結果を取得する手法と SNMP により取得する手法が考えられるが、SNMP では取得できない情報も汎用的に取得し利用できることから、コマンドを発行し結果を取得する手法を選択する。

Algorithm 1 OSPF Topology Assumption

```
1: procedure OSPF Topology Assumption
2: load Router_LSAs into RLSA_LIST
3: load Network_LSAs into NLSA_LIST
4: for all rlsa from RLSA_LIST do
5:    select link_state_id from rlsa
6:    select TRANSIT_INTERFACE_LIST from rlsa by Link_Type
7:    for all transit_interface from TRANSIT_INTEFACE_LIST do
8:       select address_of_the_interface, cost, designated_router_address
         from transit_interface
9:       select nlsa from NLSA_LIST by pairs(designated_router_address,
         address_of_the_interface)
10:       select netmask from nlsa
11:       calculate network_address from pairs(address_of_the_interface,
          netmask)
12:       output link_state_id, address_of_the_interface, network_address,
          netmask, cost, designated_router_address
13:   end for
14:   select STUB_INTERFACE_LIST from rlsa by Link_Type
15:   for all stub_interface from STUB_INTEFACE_LIST do
16:      select network_address, netmask, cost from stub_interface
17:      calculate first_address from pairs(network_address, netmask)
18:      output link_state_id, first_address, network_address, netmask,
         cost
19:   end for
20: end for
```

取得された OSPF LSDB からトポロジを推測する手法としては、OSDB に含まれる LS Type 1 のルータ LSA と LS Type 2 のネットワーク LSA をアルゴリズム 1 に示す手順で繰り返し参照し、ルータとそのルータに接続されているネットワークの関連づけを行う。このアルゴリズムの概要を以下に示す。

1. あるルータに接続されているネットワークを推測するために、ルータ LSA からトランジットネットワークに接続されているインタフェイスを探し、LSA に含まれるインタフェイスの IP アドレスとコストと指名ルータの IP アドレスを取得する。(4-8 行目)

2. その指名ルータが生成するネットワーク LSA のうち、ルータ LSA から取得したインタフェイスの IP アドレスが含まれるネットワークに関連する ネットワーク LSA を探し、含まれるインタフェイスの IP アドレスとネットマスクからネットワークアドレスを推測し、ルータと接続されているネットワークの関連づけを行う。(9-13 行目)

3. 同様に、ルータ LSA から接続されているスタブネットワークを探し、LSA に含まれるネットワークアドレスとネットマスクを収集する。その際、ルータのインタフェイスのアドレスは LSA からは取得できないため、該当スタブネットワークの先頭アドレスがルータのインタフェイスのアドレスだと推測してルータとネットワークの関連づけを行う。(14-20 行目)

```
<nodes>
  <node name="192.168.255.108">
    <interfaces>
      <interface
       name="Node192.168.255.108-Int10.0.7.0.24"
       interfaceaddress="10.0.7.2"
       netmask="255.255.255.0" ospfcost="6">
        <network name="Net10.0.7.0.24"
         networkaddress="10.0.7.0"
         netmask="255.255.255.0"/>
      </interface>
      <interface
       name="Node192.168.255.108-Int10.0.3.0.24"
       interfaceaddress="10.0.3.1"
       netmask="255.255.255.0" ospfcost="5">
        <network name="Net10.0.3.0.24"
         networkaddress="10.0.3.0"
         netmask="255.255.255.0"/>
      </interface>
      <interface
       name="Node192.168.255.108-Int10.0.6.0.24"
       interfaceaddress="10.0.6.2"
       netmask="255.255.255.0" ospfcost="7">
        <network name="Net10.0.6.0.24"
         networkaddress="10.0.6.0"
         netmask="255.255.255.0"/>
      </interface>
    </interfaces>
    <function>
      <ospf routerid="192.168.255.108">
      </ospf>
    </function>
  </node>
  ...
```

図 3: 論理ネットワークファイルの例

前述したアルゴリズムによりルータとネットワークの情報を収集しトポロジの推測を行った後、既存の AnyBed の論理ネットワークファイル形式を拡張した形式でトポロジを出力する。拡張を行った論理ネットワークファイルの例を図 3 に示す。この形式では、収集した OSPF ネットワークのルータ毎にどのネットワークインタフェイスにどのような IP アドレスが付いているか、そのインタフェイスのコストはいくつか、OSPF のルータ ID はいくつか、という情報が含まれている。

この論理ネットワークファイルに含まれる情報を元に、AnyBed の資源割り当て部が NET のクラスタ資源を割り当て、ノードの設定を生成する。ノードの設定は、大きく OS のネットワークインタフェイスやホスト名の設定と ospfd の設定に分けられる。前者の例を図 4 に、後者の例を図 5 に示す。

```
hostname 192.168.255.108-192.168.255.108
ifconfig eth1:5 inet 10.0.7.2 netmask 255.255.255.0
ifconfig eth1:2 inet 10.0.3.1 netmask 255.255.255.0
ifconfig eth1:7 inet 10.0.6.2 netmask 255.255.255.0
```

図 4: OS 設定ファイルの例

```
hostname 192.168.255.108
password sample
enable password sample
interface eth1
 ip ospf cost 5 10.0.3.1
 ip ospf cost 6 10.0.7.2
 ip ospf cost 7 10.0.6.2
 ip ospf priority 2 10.0.3.1
 ip ospf priority 5 10.0.7.2
 ip ospf priority 7 10.0.6.2
router ospf
router-id 192.168.255.108
network 10.0.7.0/24 area 0.0.0.0
network 10.0.3.0/24 area 0.0.0.0
network 10.0.6.0/24 area 0.0.0.0
redistribute bgp
```

図 5: ospfd 設定ファイルの例

3.3 実 OSPF トポロジ模倣システムの実装

実 OSPF トポロジ模倣システムを実現するため、前節までの設計を元に AnyBed のデータ収集部用に ospfwalk というスクリプトを新規に作成し、さらに既存の資源割り当て部の拡張を行った。ospfwalk でのデータの流れは図 6 のようになる。ospfwalk は実運用ルータに telnet を行ってコマンドを発行して OSPF LSDB を取得するか、又はテキストファイルとしてコマンドの出力結果を入力される。その後 LSDB から前述したアルゴリズムを用いてルータとネットワークの接続関係を推測し、論理ネットワークファイルを出力する。ospfwalk の実装は Python とグラフライブラリ networkx を利用し行った。また、論理ネットワークファイルの出力だけではトポロジを把握しにくいため、推測したトポロジを GraphViz [21] を用いて図示する機能を実装した。

AnyBed ではイーサネットを用いてトポロジを構築しているため、模倣元トポロジに含まれる Point-to-point 接続を模倣できない。このため現時点の実装では、ospfwalk において LSDB に Point-to-point 接続が存在する場合はメッセージを出してその接続を無視することとした。

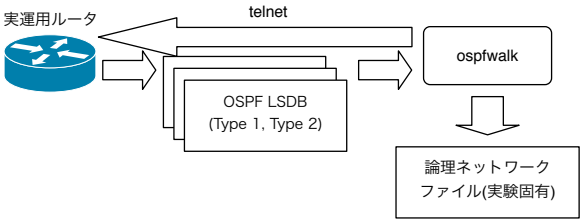

図 6: ospfwalk のデータフロー

また、既存の資源割り当て部の拡張では、既存の OSPF 設定ファイル生成機能に実アドレスの設定機能を追加した。

4 実 OSPF トポロジ模倣システムの動作検証

我々は本システムの動作検証を二段階に分けて行った。一段階目は NET 内で構築した小規模な OSPF ネットワークのトポロジを収集して模倣できるかを検証し、その後二段階目では実運用されている WIDE プロジェクトの OSPF ネットワークのトポロジを収集し模倣できるかを検証した。これら検証の詳細について次節から述べる。

4.1 小規模 OSPF ネットワークの模倣

本システムの一段階目の検証として、我々が保有する小規模 NET 内に OSPF ネットワークのトポロジ (以下 模倣元トポロジ A) を構築し、そのトポロジを収集して、再度同じ NET 内に同一のトポロジ (以下 模倣先トポロジ A) の構築を試みた。

構築を試みた模倣元トポロジ A を図 7 に示す。この模倣元トポロジ A はダンベルトポロジを元に、我々の小規模 NET の最大ノード台数に応じて複雑化したものである。すべてのネットワークには /24 のプライベート IP アドレスを割り当て、リンクのコストは模倣の検証のためばらけさせた。

検証の結果として、模倣元と模倣先で各ルータの FIB、RIB の情報を比較したところ全く同じ値となっていた。また、模倣元と模倣先の環境で図 7 に示されているルータ R1 から traceroute を他の全ルータに対して実行してみたところ、模倣元と模倣先で全く同じ経路となった。さらに、模倣元の各ルータで設定されていたインタフェイスの IP アド

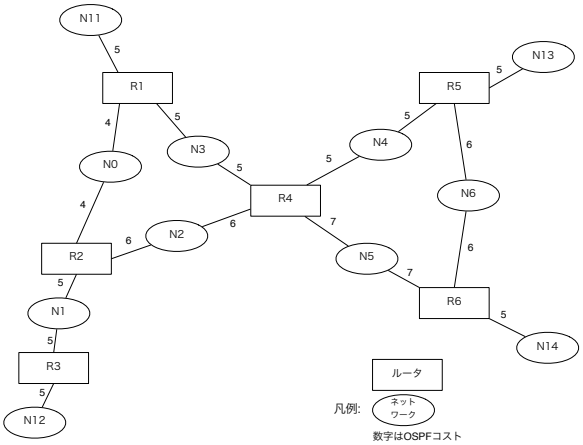

図 7: 検証用小規模ネットワーク

レスと OSPF コストは模倣先でも同じ値が設定されていることを確認した。このため本検証により、模倣元トポロジ A を再度同じ NET 上に模倣することができ、本システムは正しく小規模な OSPF トポロジを模倣できたと言える。

なお模倣先トポロジの構築に要する時間として、構築に利用する NET の OS などの準備を済ませてある前提で、模倣元トポロジの収集を開始してから、模倣先トポロジを構築し、経路が模倣先トポロジのルータすべてに行き渡って実験可能な状態になるまでの時間は計 44 秒程度であった。これは実験環境の構築に要する時間として十分に実用的であると我々は考えている。

4.2 大規模実運用 OSPF ネットワークの模倣

二段階目の検証では、実運用されている WIDE プロジェクトの OSPF ネットワークのトポロジ (以下 模倣元トポロジ B) を収集し、StarBED 上へのトポロジ (以下模倣先 トポロジ B) の模倣を試みた。WIDE プロジェクトの OSPF ネットワークは複数の OSPF エリアから構成されているが、本システムの制限からバックボーンエリアに存在するルータとネットワークの情報のみを収集した。この収集は慶応大学 SFC キャンパスに設置され GNU Zebra 0.95 [22] の動作しているサービス用ホストで行った。実験用ホストからこのサービス用ホストで動作している ospfd に直接 telnet で接続できなかったため、SSH でサービス用ホストにログインした後、telnet で ospfd のコンソールにログインし、OSPF LSDB の内容を表示するコマンドを手動で実行し結果を取得した。その後、実験用ホストに取得した結果をコピーして ospfwalk に入力した。その際、収集した模倣元トポロジ B に含まれるルータ数は 50、ネットワーク数は 253 であった。

検証の結果として、模倣元と模倣先のトポロジで同じルータに相当する数個のルータ (以下 同一ルータ) にログインし、OSPF の経路情報を確認したところ、模倣元と模倣先では経路の数が異なっていることが分かった。また、同一ルータに同じ経路が存在する場合でも OSPF コストが模倣元と模倣先で異なっている場合があった。その差異と関連してか、同一ルータから様々なルータに traceroute を行った際に模倣元トポロジと模倣先トポロジで異なる経路を通る場合があった。この原因として我々が考えているのは、前節で述べた小規模 OSPF ネットワークでの検証では正しくコストが模倣できていたことから、OSPF 以外の要素が経路に影響しているのではないかということである。例えば、個々のルータにおいて OSPF に redistribute されておらず OSPF よりも優先度の高い static 経路や connected 経路が影響している可能性が考えられる。また、ospfwalk において Point-to-point リンクを無視していることが影響している可能性も考えられる。今後、この原因についてより詳細な検討と検証を行っていくつもりである。

5 考察と今後の課題

本章では、実 OSPF トポロジ模倣システムの検証の結果得られた知見に基づき、本提案手法の課題について考察する。

5.1 OSPF トポロジ情報の収集手法について

現在の収集手法で OSPF トポロジを推測する場合、取得元のルータが属する OSPF エリア以外の情報がサマリ LSA という簡略化された形でしか取得できない。この課題は、複数エリアのルータから OSPF トポロジ情報を収集し、それらの情報をうまく繋ぎ合わせることにより解決できると考えられる。

また、現在の収集手法では模倣元トポロジのルータ上で OSPF に再配布されていない static や connected な経路は取得できない。これらの課題を解決するためには、現在の収集手法に加えて 3.2 節の手法 2 で述べたようにすべてのルータから設定と RIB / FIB を取得し、結果を綜合して模倣先トポロジに反映する必要があると考えられる。

さらに、現在の収集手法では OSPF Stub ネットワークを持つルータでそのネットワークに属するインタフェイスの IP アドレスが推測できない。これは個々のルータから SNMP やログイン後設定取得などの方法でインタフェイス情報を収集することにより解決できると考えられる。

5.2 トポロジ構築について

現在のトポロジ構築手法では、模倣元トポロジに含まれる Point-to-point 接続の模倣ができないという課題がある。これは模倣先トポロジをすべてイーサネットを用いて構築しているため Point-to-point 接続を模倣できないことによる。この課題は、模倣先トポロジ構築の際に OS に備わっているトンネルデバイスを用いて Point-to-point 接続をイーサネット上に擬似的に模倣することにより解決できると考えられる。

また、本研究では模倣元トポロジは AS 内の OSPF ネットワークのみであったが、今後は AnyBed が既に備えている BGP トポロジの構築機能を拡張し、AS 内トポロジと AS レベルトポロジを繋ぎ合わせた end-to-end のインターネットトポロジの模倣を実現していきたい。

5.3 模倣の厳密さについて

本提案手法では、模倣元トポロジのルータを PC ルータで模倣している。現在のルータはハードウェアでルーティング処理をしているものが多く、 PC ルータとの性能に違いがあると思われる。この課題を解決するためには、模倣先トポロジを構築する際に Remote Network Labs のように実機のルータを用いる手法や CISCO 7200 Simulator [23] のようなルータの動作を再現するシミュレータを用いる手法を取り入れる必要があると考えられる。

また、模倣元トポロジが物理的に広域にまたがる OSPF ネットワークの場合、厳密さを求めるためには模倣元トポロジでの各リンクの遅延を模倣先トポロジにおいても再現する必要があると思われる。これは模倣先トポロジの構築の際に各リンクに netem [24] を用いて遅延を挿入することで、ある程度解決できると考えられる。

5.4 本研究の応用について

前節で述べたように、本研究により模倣・構築された OSPF ネットワークは、模倣元ネットワークと比較するとルータの性能や各リンクの帯域・遅延を正確に模倣できていない。一方でトポロジ、各ルータの経路表、各リンクのコストについてはほぼ正確に模倣できている。このため本研究の応用先としては、ルータ性能や帯域・遅延よりもトポロジの正確性を重視する実験が向いていると考えられる。また、シミュレータと異なり、模倣したトポロジ上のルータ等においてバイナリ実装を動作させられることも本研究の利点である。これらのことから、我々が考えている応用先としてネットワークトポロジに依存するバイナリ実装の評価、ネットワーク設計、ネットワーク障害時の挙動調査などがある。しかし残念ながら本研究が実際に応用された事例はまだ存在しない。

6 おわりに

本稿では、AS 内ネットワークの OSPF トポロジを NET 上に模倣し、そのトポロジ上で実験・検証・教育などが行える、実 OSPF トポロジ模倣システムとそのシステムによる模倣ネットワーク構築の試行と結果について述べた。試行の結果、小規模な OSPF トポロジの模倣には成功したが、大規模な実運用トポロジの模倣には本提案システムのアプローチでは限界があることが分かった。今後は、OSPF トポロジ模倣の再現性の向上と共に、AS 内 OSPF ネットワークと AS レベル BGP ネットワークを組み合わせて模倣し、ある AS 内のネットワークから上位の AS を経由し別の AS 内ネットワークまでの end-to-end のネットワークトポロジの模倣に取り組んでいくつもりである。

参考文献

[1] NICT Hokuriku Research Center. http://starbed.nict.go.jp/.

[2] Toshiyuki Miyachi, Kenichi Chinen, and Yoichi Shinoda. StarBED and SpringOS: Large-scale General Purpose Network Testbed and Supporting Software. In *Proceedings of International Conference on Performance Evaluation Methodlogies and Tools (Valuetools) 2006*, October 2006.

[3] Mio Suzuki, Hiroaki Hazeyama, Daisuke Miyamoto, Shinsuke MIWA, and Youki Kadobayashi. Expediting Experiments Across Testbeds with Anybed: A Testbed-Independent Topology Configuration System and Its Tool Set. *IEICE Transactions on Information and System*, Vol. E92-D, No. 10, pp. 1877–1887, October 2009.

[4] Shinsuke Miwa, Mio Suzuki, Hiroaki Hazeyama, Satoshi Uda, Toshiyuki Miyachi, Youki Kadobayashi, and Yoichi Shinoda. Experiences in emulating 10K AS topology with massive VM multiplexing. In *Proceedings of The First ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures(VISA 2009)*, August 2009.

[5] Route Views. University of Oregon Route Views Project. http://www.routeviews.org/.

[6] CAIDA: The Cooperative Association for Internet Data Analysis. The CAIDA AS Relationships Dataset. http://www.caida.org/data/active/as-relationships/.

[7] X. Dimitropoulos et.al. AS Relationships: Inference and Validation. *ACM SIGCOMM Computer Communication Review (CCR)*, Vol. 37, No. 1, pp. 29–40, 2007.

[8] T. Benzel et.al. Design, Deployment, and Use of the DETER Testbed. In *Proceedings of DETER community workshop 2007*, Aug. 2007.

[9] Huan Liu and Dan Orban. Remote Network Labs: An On-Demand Network Cloud for Configuration Testing. In *Proceedings of Workshop: Research on Enterprise Networking (WREN 2009)*, August 2009.

[10] Information Sciences Institute. NS-2 network simulator. Software Package, 2003. http://www.isi.edu/nsnam/ns/.

[11] Ssfnet: Scalable simulation framework - network models. http://www.ssfnet.org. See http://www.ssfnet.org/publications.html for their publications.

[12] 小原泰弘, 南政樹, 中村修, 村井純. ルーティングシミュレータ simrouting の開発. マルチメディア分散, 協調とモバイルシンポジウム (DICOMO2007) 論文集, pp. 205–211, July 2007.

[13] Jared Winick. Inet Topology Generator. http://topology.eecs.umich.edu/inet/.

[14] E. W. Zegura. GT-ITM: Georgia Tech Internetwork Topology Models, 1996. http://www.cc.gatech.edu/fac/Ellen.Zegura/gt-itm.

[15] Alberto Medina and Anukool Lakhina and Ibrahim Matta and John Byers. BRITE: An Approach to Universal Topology Generation. *In Proceedings of the International Workshop on Modeling, Analysis and Simulation of Computer and Telecommunications Systems(MASCOTS 2001)*, August 2001.

[16] Rocketfuel. An isp topology mapping engine. http://www.cs.washington.edu/research/networking/rocketfuel/.

[17] Roman Chertov and Sonia Fahmy and Pankaj Kumar and David Bettis and Abdallah Khreishah and Ness B. Shroff. Topology Generation, Instrumentation, and Experimental Control Tools for Emulation Testbeds. In *Proceedings of DETER community workshop 2006*, June. 2006.

[18] OSPF network visualizer: ospfviz. http://ospfviz.sourceforge.net/index.html.

[19] 兒玉清幸, 釜崎正吾, 吉田和幸. ネットワーク構成情報表示システムのための自動配置アルゴリズムの評価. マルチメディア分散, 協調とモバイルシンポジウム (DICOMO2007) 論文集, pp. 1754–1761, July 2007.

[20] Quagga Software Routing Suite. http://www.quagga.net/.

[21] Graphviz - Graph Visualization Software. http://www.graphviz.org/.

[22] GNU Zebra. http://www.zebra.org/.

[23] C. Fillot. Cisco 7200 simulator. http://www.ipflow.utc.fr/index.php/Cisco_7200_Simulator.

[24] S. Hemminger et.al. Network Emulation with NetEm. In *Proceedings of Linux Conf Au 2005*, April 2005.