2011年度 Deep Space One / Nerdbox Freaks Working Group 活動報告

榎本真俊 (masatoshi-e@is.naist.jp)　太田悟史 (sota@nict.go.jp)
知念賢一 (k-chinen@jaist.ac.jp)　櫨山寛章 (hiroa-ha@is.naist.jp)
宮地利幸 (miyachi@nict.go.jp)　宮本大輔 (daisu-mi@nc.u-tokyo.ac.jp)
安田真悟 (s-yasuda@jaist.ac.jp)　三輪信介 (danna@nict.go.jp)

2011 年 12 月 31 日

目 次

1 はじめに

ここでは，実ノードを用いた大規模なインターネットシミュレーション環境の構築の研究開発を行っている Deep Space One WG および Nerdbox Freaks WG の活動報告を行う．Deep Space One WG は実環境向けのハードウェアおよびソフトウェアを利用した大規模な実験用環境の構築・運用に関する研究に取り組み，Nerdbox Freaks WG では大規模実験環境のユーザ視点から利用方法やノウハウの共有，実験例，新たな利用例の考案，実験・開発ワークショップを行っている．

今年度から Deep Space One WG は新しく榎本，太田をチェアとした活動を開始した．新チェアのもとでは新たに研究トピックとして「テストベッド連携」,「大規模実験環境制御 API による実験環境構築の自動化・効率化」,「大規模かつ複雑な実験環境構築の迅速化」に注力して研究開発を進めていく．

一方，Nerdbox Freaks WG では従来までの大規模実験環境の利用促進や開発ワークショップの開催，インターネットエミュレーションを用いた実験環境構築手法の研究開発に加え，テストベッド環境を用いた実験手法の確立や，利用者視点によるマニュアルや細かな設定に関する知見などを効果的に実験利用者間で共有する実験環境構築におけるナレッジベースの作成を新たに実施していく．

以降 Deep Space One WG 及び Nerdbox Freaks WG の本年度の主な活動について報告する．

1.1 用語定義

まず本報告で用いる技術用語の定義を行う．

物理マシン：
物理的なサーバ資源やルータ，スイッチ等，物理的な機械を示す．

仮想マシン：
XEN，VMWare，KVM 等が生成した仮想的なサーバ資源やルータ，スイッチ等，仮想的な機械を示す．

仮想ノード：
仮想マシン上に作成されたノードを示す．

物理ノード：
物理マシン上に作成されたノードを示す．

実験ノード：
仮想ノード，物理ノードを問わず，実験に利用するノードを示す．

AS：
Autonomous System（自律システム）の略語である．一般的にインターネットにおけるネットワーク運用管理単位として用いられる．

模倣インターネット：
実インターネット上で観測されたデータセットをもとにテストベッド上にエミュレーション技術を用いて構築した，インターネットを模したネットワークを示す．模倣環境と呼称する場合もある．

BGP 網エミュレーション環境：
模倣インターネットの形式のひとつであり，実インターネットの BGP 網を模してテストベッド上に構築したエミュレーション環境を示す．

P2P 網エミュレーション環境：
模倣インターネットの形式のひとつであり，実インターネットで利用されている P2P アプリケーションを利用してテストベッド上に構築した P2P 網のエミュレーション環境を示す．

2 StarBED/SpringOS のアーキテクチャと性能評価

StarBED は 2002 年に設置されてからノードの増減および，その効率の検討などからアップデートが繰り返されてきた．また，StarBED での実験を支援する為の SpringOS もその開発途上で新たなモジュールの追加や統合などによりそのアーキテクチャを変更している．

本節では，2010 年時点の StarBED および SpringOS のアーキテクチャおよびその性能評価結果についてまとめる．

2.1 StarBED と SpringOS のアーキテクチャ

まず StarBED と SpringOS の概観とアーキテクチャについてまとめる．

2.1.1 StarBED の L2 トポロジ

StarBED の L2 トポロジを図 1 に示す．StarBED に設置されている実験用ノードは実験用と管理用のネットワークに接続されるそれぞれのネットワークインターフェースを持っている．管理用ネットワークは静的に設定されており，実験中や環境の整備段階であっても常にノードへのネットワーク到達性を提供する．実験用ネットワークはこの管理用ネットワークを用いて実験用ノードを設定し，実験用ネットワーク側に必要な実験トポロジを構築する．StarBED では VLAN 設定

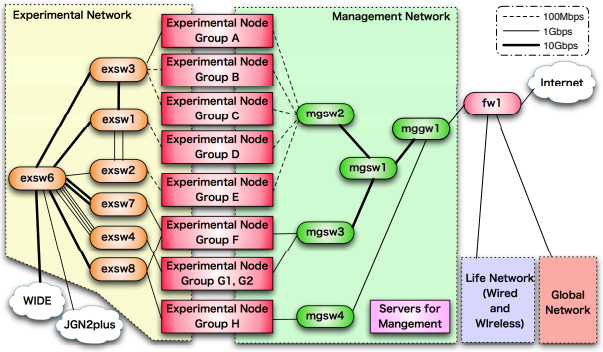
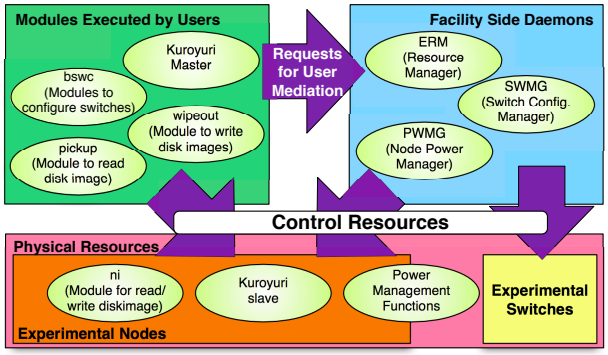


図 1: StarBED の L2 トポロジの概念図



Kuroyuri slave execute commands sent by Kuroyuri master to drive experimental scenario, but Kuroyuri master also has functions of wipeout and bswc.

図 2: SpringOS のモジュール群

による L2 トポロジの構成変更機能が提供されており，L3 設定に関しては実験ノードの設定およびソフトウェアルーターの導入などにより実験実行者が行う．実験ノードが接続されているこれらの 2 つのネットワークは，インターネットや StarBED 内の業務用ネットワークなどからは分離されており，利用者は踏み台ノードを経由して接続を行う．ただし，必要に応じて WIDE や JGN を経由しての外部接続が可能である．

2.1.2 StarBED の実験用ノード

StarBED に設置されている実験用ノードは，その導入年度などにより，スペックが異なり，スペック毎にグループ A から H に分類されている．特にネットワークインターフェース数も異なるため実験により利用するノードを利用者が選択する．ノードの仕様を表 1 に示す．

2.1.3 SpringOS の機能

SpringOS は主に StarBED 上での実験を支援する為のソフトウェアスイーツであり，複数のモジュールから構成される．StarBED の主な機能を以下に挙げる．

リソース管理 StarBED に存在するノードや VLAN の情報を管理し，さらに実験実行者の要求によりノードの割り当てを行う．

ノードの死活管理 IPMI や WoL，SNMP などを利用してノードの電源投入および電源断，再起動などを行う．

ノードへのソフトウェア導入 OS の導入やさまざまなソフトウェア設定された一台の実験ノードのハードディスクの内容をディスクイメージとして保存し，それを複数台の同一仕様の実験ノードに配布することでノードの複製を行う．

L2 トポロジの構成 実験ネットワーク上に存在するスイッチの VLAN を設定することで L2 トポロジを構成する．

シナリオ実行 OS の導入などが終了した実験ノードで，前もって実験実行者により用意されたシナリオファイルにしたがって，必要なプログラムなどを実行して実験を遂行する．メッセージ交換によるノード間協調機能も提供している．

2.1.4 SpringOS を構成するモジュール郡

前述の通り SpringOS は多数のモジュールから構成されている．これらには，ネットワークテストベッドに存在する共用リソースを制御するものや，実験ノード上で動作し，ノードの挙動を制御するもの，そして，これら 2 種類のモジュールに対するユーザインターフェースを提供するモジュール郡の 3 種類が存在する．図 2 に主なモジュールとその役割を示す．

表 1: StarBED の実験ノード (2010 年 3 月現在)

		A	B	C	D	E	F	G1	G2	H
Group		NEC						Proside		HP
Model		Express5800						AmazeBlast		ProLiant
		110Rc-1		120Ra-1	110Rc-1		110Rg-1	neo920		DL320 G5p
CPU		Pentium3 1GHz					Pentium4 3.2GHz	Opteron 2GHz		Xeon X3350
Memory		512MB					8GB	8GB	4GB	8GB
Disk		IDE 30GB		SCSI 36GB	IDE 30GB		SATA 80GB * 2	unavailable		SATA 160GB
Exp. NIC	FE	0	1	4	1	4	0	0	0	0
	GbE	1	0	0	0	0	4	1	1	2
Mmgt NIC	FE	1	1	1	1	1	0	0	0	0
	GbE	0	0	0	0	0	1	1	1	1
Node no.		208	64	32	144	64	168	100	50	240
Introduced in		2002					2006	2007		2009

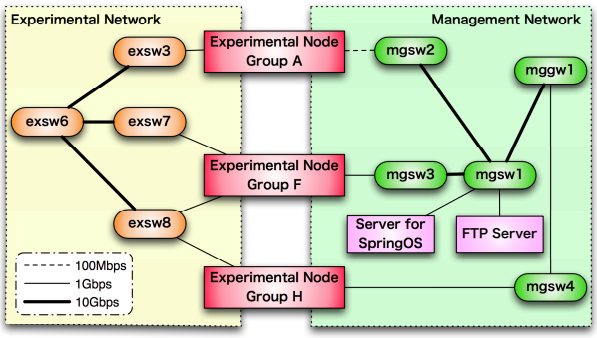


図 3: 実験に利用した環境

2.2 性能評価

StarBED 上での SpringOS の性能を評価するための実験を行った．今回は特に，ソフトウェアの導入のために必要な時間，スイッチの設定に要する時間，そしてシナリオ実行の精度について計測を行った．

利用した実験環境を図 3 に図示した．対象とするノードグループは仕様などからグループ A，F，H の 3 グループとした．ディスクイメージ配布のために利用した FTP サーバと，SpringOS の施設向けモジュール群が動作するノードを表 2 に示す．

2.2.1 ソフトウェア導入

ソフトウェア導入に必要な時間を計測するために 3 種類の計測を行った．

実験 A1 実験ノードに配布するテンプレートとなるディスクイメージの作成時間の計測．ただし，zerofree というプログラムを利用してディスク上の利用していない inode を”0” で上書きしたディスクイメージと zerofree を実行しなかった二種類のディスクイメージの作成時間をそれぞれ計測した．

実験 A2 A1 で作成したディスクイメージを複数台のノードに配布するための要する時間の計測

実験 A3 時間経過と書き込んだディスク容量の関係の観測．ただしここでは 3 種類のディスクイメージを用意した．一つ目は SpringOS の標準である zlib を利用して圧縮したディスクイメージ (以下 default)，二つ目は zerofree を実行し，さらに zlib で圧縮したもの (以下 zerofree)，そして三つ目は zerofree は実行したが圧縮をしていないもの (以下 uncompress) であり，これらのディスクイメージを 1 台のノードに配布した．

実験に利用したディスクイメージのもととなったハードディスクには，20GB のパーティションを作成し，

表 2: 実験に利用したサーバ群の仕様

Role	Model	CPU	Memory	OS
SpringOS server	HP ProLiant DL 360 G5	Xeon E5405	2GB	Solaris 10
FTP Server	HP ProLiant DL 380 G5	Xeon E5405	16GB	Solaris 10

表 3: 実験 A1 の結果

Group	without zerofree		with zerofree		Image size Ratio [%]
	Creation Time [sec]	Size [GB]	Creation Time [sec]	Size [GB]	
A	6821	9.1	3551	0.760	8.35
F	3569	6.3	2687	0.875	13.89
H	2875	2.4	2539	0.870	36.25

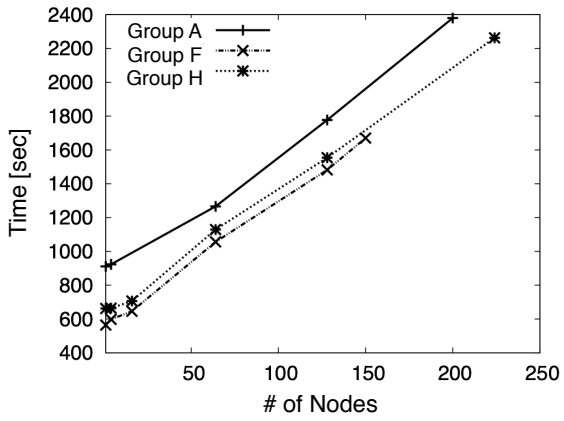


図 4: 実験 A2 の結果

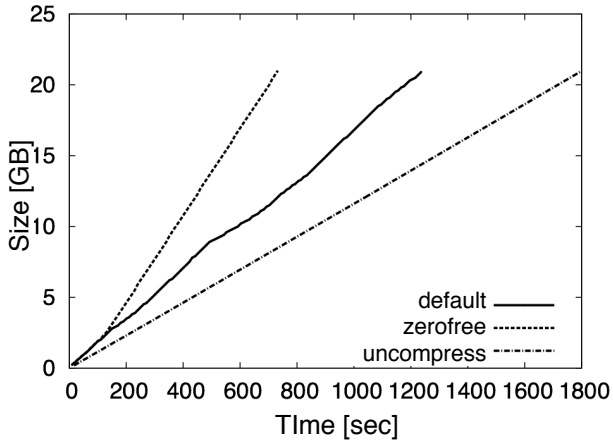


図 5: 実験 A3 の結果 (Group A)

Fedora 13 を標準設定インストールした．

実験 A1 の結果を表 3 に示す．zerofree を利用すると，圧縮効率が高まるため，保存されたディスクイメージのサイズを大幅に押さえられることが観測できた．割合は対象としたノードが以前どのような使われ方をしていたかに依存するためそれぞれのグループで差がみられたと考えられる．

実験 A2 の結果を図 4 に示す．Group A のパフォーマンスが低いのは他のノードに比べてスペック的に劣るためであると考えられる．

実験 A3 の結果を図 5 および図 6 に示す．Group A の性能が Group F より低い原因は Group A の管理側ネットワークインターフェースが FastEthernet のものであることと考えられる．また，Group A の zerofree を利用したディスクイメージでは 3GB 以下とそれ以降のグラフ傾きが変わっている．これは，この 3GB の部分に LinuxOS の実体が保存されており，zlib による圧縮の効率が 0 で埋められている部分と比較して悪い事が原因であると考えられる．また，default のグラフの傾きが安定しないのはディスク上の位置ごとに圧縮効率が変化しているためである．

2.2.2 スイッチ設定

スイッチ設定に関しては，以下の実験を行った．

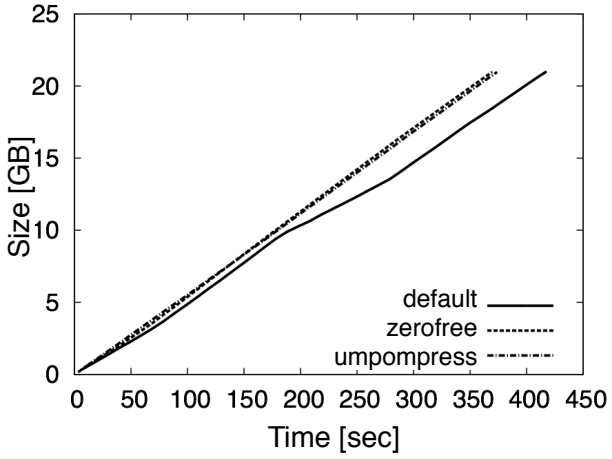


図 6: 実験 A3 の結果 (Group F)

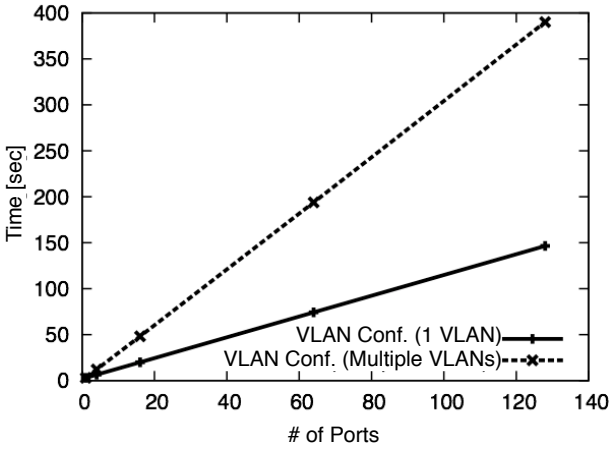


図 7: 実験 B1 および B2 の結果

実験 B1 一つの VLAN に複数のポートを所属させる設定を行った場合に必要となる時間

実験 B2 一つのポートのみが所属する VLAN を複数個作成する際に必要となった時間

実験 B1 の結果を図 7 に示す．SpringOS が VLAN 作成時にスイッチ上で発行するコマンドの実行時間により，多数の VLAN を生成する場合の必要時間が比較的長くなっていることが分かる．

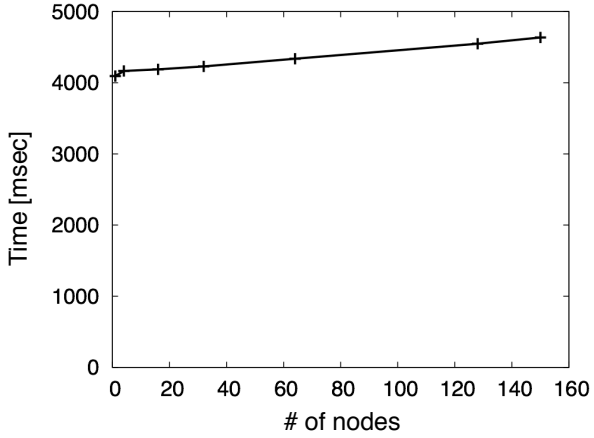


図 8: 実験 C の結果

2.2.3 シナリオの遂行

シナリオの実行に関しては，実験 C として，以下のシナリオ実行に必要となった時間を計測した．

1. kuroyuri master から，実験ノードで動作している kuroyuri slave に IP アドレスを含むメッセージを送信
2. このメッセージをトリガとして kuroyuri slave がメッセージに含まれていた IP アドレスに対して ping コマンドを利用して 5 つの ICMP Echo Request を送信
3. ping コマンド終了後，kuroyuri slave は kuroyuri master に終了を示すメッセージを送信

図 8 に実験 C の結果を示す．ノード数賀 150 台でも 1 台のノードと比較して 500ms のみの増加が観測されており，そのオーバーヘッドはそう大きくないといえる．また，手動で本シナリオが実行する ping コマンドを実行した際には平均 4030ms の RTT が観測されており，シナリオ実行によるオーバーヘッドも小さいといえる．

2.3 まとめ

SpringOS を利用して，多数のノードを制御する場合，ディスクイメージの導入は，一台当たり 10 秒台，1 つのスイッチポートを VLAN に所属させる設定は数

秒という短い時間で実行できることが確認できた．また，ディスクイメージ作成時には zerofree を実行することで，保存するディスクスペースおよび，書き込み時間の双方に効果があることが確認できた．

現在は，StarBED の実験ノード構成も更新されており，表 1 に示したノードののうち A-E を廃棄し，新たなノードを導入した．新たな実験ノードとしては Cisco 社の UCS C200 M2(Intel 6-core Xeon X5670 x 2, 48 GB Mem, SATA 500GB x 2 HDD, 4 GbE for experiments) が 552 台導入されており，今後これらのノードを利用して同様の性能評価実験を行う予定である．本節の内容は wide-paper-deepspace1-tridentcom2011-00.txt が詳しい．

3 実験環境構築の効率化

実験環境構築の効率化に関する研究開発として，大規模実験環境への OS イメージ投入の高速化における技術要件の整理と，FAI (Fully Automation Install) を利用した StarBED 実験ノードへの OS インストール手法の開発に今年度は取り組んだ．

3.1 大規模実験環境への OS イメージ投入の高速化における技術要件の整理

大規模ネットワーク実験環境での実験支援ソフトウェアとして，StarBED では SpringOS を開発運用している．しかし，StarBED のノード台数の増加，実験者が要求する実験規模の拡大に対して SpringOS は十分な信頼性，規模追従性を有していない．その為，実験者が大規模な実験環境を構築する際に，様々な障害に遭遇する．

これまでは，これまでの運用で培ったノウハウからそれらの障害の回避や復旧を行っていたが，今回その原因の解析，対処方法の整理を行い，耐規模性を持ったネットワーク実験支援ソフトウェアに必要な技術要件を調査した．その結果，大規模な実験環境を構築する際に障害を発生させる原因は以下の 3 つである事が明らかになった．

- Remote Management Interface の信頼性の不足
- Management Server の過負荷
- Management Network の輻輳

以下に各障害原因の詳細を述べる．

- **Remote Management Interface**
  StarBED の実験ノードは外部から電源制御・監視を行う為に WoL, IPMI, ILO 等の Remote Management Interface を持っている．しかし，これらは通信に UDP を用いていて，多数のノードに制御メッセージを送信した場合にパケットロスにより制御が行われない場合がある．また，頻繁な制御を行うと Interface Board 自体の動作が不安定になる場合もあった．

- **Management Server の過負荷**
  大規模な実験環境を構築する際には，実験で用いるノードのディスクイメージや，設定ファイルの配布するためのファイルサーバ (FTP/TFTP/NFS) や，DHCP サーバを利用する事になる．しかし，これらのマネジメントに用いるサーバノードは数台程度であり，大規模な実験環境を構築する際に，サービスリクエストが集中すると障害が発生する場合がある．

- **Management Network の輻輳**
  Management Server の過負荷と同様であるが，大規模実験環境を構築する際には，大量の通信が発生する．その際にマネジメントサーバが正常に動作していても，マネジメントネットワークが輻輳することで，環境構築の障害となる場合がある．

これらの知見を踏まえ，規模追従性，及び信頼性の高い実験支援ソフトウェアを作成する為に，我々は実験支援ソフトウェアが持つべき機能・技術要件を提案した．詳細は European Conference of Computer Science 2011 (ECCS'11) にて発表した論文 wide-paper-deepspace1-simabuki-eccs2011-00.txt を参照してほしい．

3.2 FAI を用いた StarBED での実験環境構築

前節で述べたとおり，Spring OS を用いたディスクインストールはディスクイメージが大きいほど，または，インストールするノードの数が多いほど時間を要

し，実験ノードが 10 台程度であれば CD などのインストールイメージで OS インストールをする方が短時間でセットアップできる場合もある．一方，一般的な CD インストールイメージからの OS インストールは，リモートコンソールなどでコマンド入力を行いながら設定しなければならず，StarBED のような大規模な HaaS 実験環境への OS インストールは時間と手間を費やす作業となる．

そこで，FAI (Fully Automatic Installation) [7] を用いて StarBED の実験ノードにベース OS イメージを自動的にインストールする手法の開発を行った．FAI 設定の参考資料として東京大学 田浦 健次朗氏の wiki [8] に FAI の基本的な設定方法がまとまっている．田浦氏のメモを元に，StarBED 用にいくつか変更を加えた点をここにまとめる．

今回用いた OS は Debian 6.0.4 x86_64 イメージである．以下手順を説明する。

手順 0. FAI サーバの用意

通常の利用では，実験ノードのうちの一台を利用して作成する．FAI サーバは基本的には他の実験ノードと同じ OS イメージで構築するのが無難である．

手順 1. fai-quickstart のインストール

Debian の場合，図 9 のように apt からインストールできる．

手順 2. NFS サーバ，DHCP サーバの設定

次に NFS サーバ，DHCP サーバの設定を行う．StarBED 環境では NFS サーバは基本的には手順 0 で用意した FAI サーバ上で図 10 の内容を/etc/exports に設定し nfs-kern-server を起動する．

StarBED 環境では実験ノード用の DHCP サーバは用意されているので，StarBED の DHCP サーバを用いる．DHCP サーバ側の設定変更は特にない．

手順 3. fai-setup による FAI の初期設定

用意した FAI サーバにて，図 11 に示す fai-setup コマンドを実行し FAI の初期設定を行う．特に変更をしなければ FAI の root ディレクトリは /srv/fai/ になる．

手順 4. FAI 設定ファイルのコピー

次に，図 12 に示すコマンドで FAI の設定ファイルを /src/fai/config 以下にコピーする．

手順 5. FAI 用の kernel と initrd の編集

StarBED の実験ノードに CD インストールイメージでネットワークインストールを行う場合，HTTP プロキシを設定しておかなければならない．そのため，StarBED 管理者に FAI インストールするノードに対する HTTP プロキシの設定を有効にしてもらった上で，initrd の中に含まれる init/scripts/live というスクリプトに HTTP プロキシの設定を書きくわえる必要がある．まずは適切な作業ディレクトリに移動し，図 13 のコマンドで initrd のイメージを展開する．

次に展開した initrd の中に含まれる init/scripts/live というスクリプトに HTTP プロキシの設定を書きくわえる．図 14 はグループ L のノードに対する HTTP プロキシの設定を加えた init/scripts/live と元々の init/scripts/live の差分である．編集した initrd は図 15 のコマンドで圧縮する．

手順 6. initramfs の修正

また，initramfs にも修正を加える必要がある．まず必要となるデバイスドライバをインストールする。StarBED の CISCO UCS サーバを利用する場合は初期設定で入っている Intel のデバイスドライバの他に Broadcom のデバイスドライバ (bnx2-firmware) をインストールする。debian の場合、non-free パッケージに bnx2-firmware は含まれているため、/etc/apt/apt.conf に non-free を追加して apt にてインストールする。

次に、initramfs-tools を apt でインストールする。その後、/etc/initramfs-tools/modules に “bnx2” を追加し、図 16 のコマンドで initramfs を作成する。

手順 7. FAI 用 kernel と initrd の mgsv1 へのコピー

StarBED で PXEBoot を行う際, kernel イメージ，initrd イメージ，Linux の場合は pxelinux.cfg を mgsv1 の/tftpboot/ ディレクトリに設置する必要がある．/tftpboot/ ディレクトリへの PXEBoot

```
apt-get install fai-quickstart
```

図 9: apt による fai-quickstart のインストール

```
/srv/fai/config 172.16.0.0/255.255.0.0(async,ro,no_subtree_check)
/srv/fai/nfsroot 172.16.0.0/255.255.0.0(async,ro,no_subtree_check,no_root_squash)
```

図 10: FAI 用 /etc/exports の例

```
fai-setup -v
```

図 11: fai-setup のコマンド実行

```
cp -a /usr/share/doc/fai-doc/examples/simple/* /srv/fai/config/
```

図 12: FAI 設定ファイルのコピー

```
zcat /boot/initrd-2.6.xx.x.img | cpio -id
```

図 13: zcat による initrd の展開

```
root@hokuriku-go:/xenebula_root/tftproot/init/scripts# diff live live.orig
6d5
< export http_proxy="http://172.16.24.248:8080"
```

図 14: HTTP プロキシの設定を加えた init/scripts/live の diff による差分

```
find . | cpio -o -H newc | gzip > /boot/initrd-2.6.xx.x.img
```

図 15: gzip による initrd の圧縮

```
update-initramfs -k 2.6.32-5-amd64 -c
```

図 16: update-initramfs による initramfs の作成

起動イメージの設定ツールとしてSpring OSの一つであるkiyomitsuというWebインターフェースを備えたツールが開発されており，近い将来正式なリリース予定であるが，ここでは，旧来通りの手動での設定を説明する．

まずmgsv1の/tftpboot/以下にStarBED利用プロジェクト名などでディレクトリを作成する．そこにkernelイメージ（vmlinuz），initrd，pxelinux.cfgをFAIサーバからコピーする．ここでは，kernelイメージとinitrdはmgsv1:/tftpboot/naist-anybed/以下にpxelinux.cfgはmgsv1:/tftpboot/naist-anybed/pxelinux.cfg/にdefault-autoinstallという名前で設定したものとする．

手順 8．mgsv1 での pxelinux.cfg の設定

図17はpxelinux.cfgの例である．172.16.24.204は今回用意したFAIサーバのグループKにおけるアドレスである．

つぎに，図18のようにmgsv1の/tftpboot/pxelinux.cfg以下に利用する実験ノードのマネジメント用IPv4アドレス16進数表記のファイル名で設置したpxelinux.cfgへのシンボリックリンクを設定する．図18はグループKのノードk002 (IPv4アドレス172.16.24.2)に対して/tftpboot/naist-anybed/default-autoinstallへのシンボリックリンクが設定された状態を示す．

また，64bit OSイメージをインストールする場合は，mgsv1:/tftpboot/以下に利用する実験ノードのpxeリンク（k002.pxeなど）を削除しておく．32bit OSの場合はSpring OSのsbpshにて設置したpxeliunx.cfgへのリンクを設定しておく．

手順 9．実験ノードの起動と FAI 実行

FAIでOSインストールを行う実験ノードを起動すると，実験ノードはPXEBoot経由でFAIサーバのNFSディレクトリからFAIのインストーラーを読み込み，インストールを開始する．この際初回のインストールは必ず失敗するため，インストールの失敗を確認したのち，再起動してインストールを再実行する．

インストールが終了した後，再びインストールを行わないために，手順7で行ったmgsv1上のpxelinux.cfgへのシンボリックリンクの設定を削除しておく．そして実験ノードを再起動させ，ディスクへインストールしたOSイメージで実験ノードが起動してくることを確認する．

現時点では手順8にてインストール実行が2回必要となっている．何故失敗するのかの原因はまだ明らかではないが，今後改善していく．また，FreeBSDでもsysinstallを用いて同様の事が行えるため，FreeBSDでのFAIによるStarBED実験ノードへのOSインストール手法の開発も今後進めていく．

4 実験環境構築の高度化

実験環境構築の高度化に関する研究開発として，実験環境への障害導入手法の研究開発と実験環境へのフィッシングサイト再現手法の研究開発に今年度は取り組んだ．

4.1 実験環境への障害導入

インターネットを初めとする実環境では常に何らかの障害が発生しているといっても過言ではない．現在我々が利用しているインターネットはさまざまな技術が複雑に連携しながら，その「健全性」を確保している一方，一見全く関連がないとおもわれるような技術の障害がさまざまなその他のサービスに影響を与える可能性がある．我々はこれまで現実的な実験用環境の構築に取り組んできたが，多くの場合，対象とする技術に直接関係するような障害を意図的に入れることはあれど，それ以外の要素に関しては健全なものを想定してきた．技術の検証段階でインターネット上に存在するさまざまな要素による，検証対象技術への影響を調査し，実環境への導入後に起こりえる問題を把握しておくことは重要である．

我々はこれまで開発・運用してきた技術を応用し，主に実環境向けの実装を用いたネットワーク技術向けの検証環境に障害を入れるためのフレームワークを提案し，さらにいくつかのケーススタディ実験を行い我々のフレームワークの有効性を示した．

```
default fai-generated

label fai-generated
kernel naist-anybed/vmlinuz-2.6.26-2-amd64
append initrd=naist-anybed/initrd.img-2.6.26-2-amd64-v6 ip=dhcp \
root=/dev/nfs nfsroot=172.16.24.204:/srv/fai/nfsroot boot=live \
FAI_FLAGS=verbose,sshd,createvt FAI_ACTION=install
```

図 17: FAI 用 pxelinux.cfg の例

```
guest@mgsv1 % ln -s /tftpboot/pxelinux.cfg/AC101802

lrwxrwxrwx 1 guest guest 41 Feb 1 11:09 AC101802 ->
../naist-anybed/pxelinux.cfg/default-autoinstall
```

図 18: FAI 用 pxelinux.cfg へのシンボリックリンク

4.1.1 ネットワーク障害

ネットワーク障害とは PC やネットワーク機器やそれらを接続する物理リンクなどネットワークを構成する全ての要素で起きうる障害であり，実ネットワーク上では日常的に発生している．これらの障害はある点から見た際には，発生した場所やその種類などによりさまざまな挙動を見せる．また，観測する場所の違いによってもその挙動は当然異なる．

4.1.2 我々のアプローチ

前節で述べた障害の種類や観測地点を整理し，これらを既存の手法で実験用環境に導入する．ここで利用する既存の手法は以下の 3 種である．

実ノードによる環境

実環境で利用されているものと同等の機器やソフトウェアを利用して実験用環境を構築する．本手法では実環境と同等の障害を発生させることが可能でありその伝搬の様子の観測も可能である．しかし，その一方構築の為のさまざまなコストは大きい．さらに，物理的に実験機材を破壊するような障害については一般的には導入する事は不可能である．

ソフトウェアによるシミュレーション

障害をモデル化し，ある問題が起きた際にその影響がどのように広がっていくのかを適切に模倣することで小規模なリソースで障害の再現が可能であるが，障害のモデル化を正確に行えていない場合は，実現した環境の現実性に問題が起こることもある．

仮想ノードを利用した環境

仮想化技術の高度化と，手軽に豊富なハードウェアリソースの利用が可能となってきており，一台の物理ノード上で複数のノードを動作させる仮想ノードの利用は今はあたりまえともいえる状況である．これを利用する事により環境の大規模化がはかれる．また，仮想ノードは物理ノードのハードウェア部分をシミュレートしているため，その上で動作する実環境用のソフトウェアに対して，ハードウェア部分での障害を擬似的に導入する事

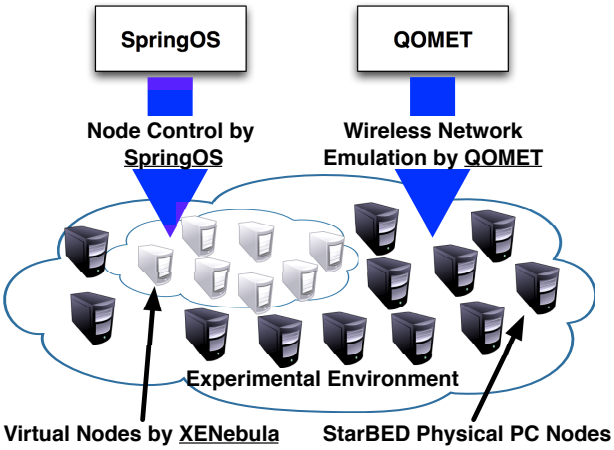


図 19: 我々のアプローチ

も理論的には可能である．

これらの 3 種の技術を実験用環境の適所に配置することで，柔軟かつ現実的な環境を構築する．我々はこれまで，実験を容易に行う為のツール群を作成してきた．本アプローチではこれらを統合して用いる．

StarBED は多数の物理リソースを有するネットワークテストベッドであり，StarBED を利用すれば多数の実ノードを実験環境に導入する事が可能である．実験環境の規模が足りない場合などは XENebula を利用する事で多数の仮想ノードの設定および起動を行う．また，QOMET を障害のシミュレーションに利用する．QOMET は，本来無線リンクを有線リンク上に模倣する技術であり，無線フィールドのモデルにしたがい，無線ノード間のリンク特性を再現する．QOMET が持つ無線フィールドに関するモデルの部分を障害モデルに差し替えることで障害のシミュレーションが可能となる．また，これらの実験用ノードを設定・制御する為に SpringOS を利用する．本アプローチの概念図を図 19 に示す．

4.1.3 ケーススタディ

本アプローチを利用したケーススタディを 2 例行った．

一つは，Home Energy Management System (HEMS) に関するものであり，障害が発生する環境と発生しない環境の 2 種類を用意し，各家庭の消費電力にどのような影響があるかを比較した．これにより，障害が発生する環境では，電力サーバが各家庭の電力消費量を制御できないケースが発生しその時間帯には，電力消費量が増大することが観測できた．

もう一つのケーススタディは，Interop 2011 の開催者イベントとして行われた Cloud Computing Competition において参加者に提供した環境であり，この環境を利用して参加者はそれぞれが提案する技術のライブデモンストレーションを行った．本環境では参加者が手動で障害をいれるインターフェースを提供し，参加者による障害時の提案技術の挙動についてのデモンストレーションが可能となった．

4.1.4 今後の予定

今回のケーススタディで利用したシミュレーションモデルはもともと QOMET が持つ無線リンクのためのものであり，障害向けのものではない．今後は障害のモデリングについての検討を行う．また，本アプローチを使うためのインターフェースは用意されておらず，それぞれのツールを別々のインターフェースで制御して環境を構築している．実験者が容易に本提案による障害導入を行うためには統一されたインターフェースの開発が必要である．

本提案のアプローチおよびケーススタディの詳細については，wide-paper-deepspace1-mya-aintec2011-00.txt で詳しく説明している．

4.2 模倣インターネットにおけるフィッシングサイトの再現

インターネットの AS 間隣接構造の再現に端を発した模倣インターネットは，DNS サーバ構造など様々なシステムの再現研究が行われている．本研究では，ウェブサイト，とりわけエンドユーザを騙すことによって個人情報を盗み取る犯罪行為であるフィッシングサイトの再現を試みる．

これまでフィッシングサイトの対策として，そのサイトの真贋判定を行うようないわゆる検知システムが開発され続けている．しかし，フィッシングサイトは，サイトが作成されてから消滅するまでの期間が短いことが知られており，検知システムを評価するのに十分な量のフィッシングサイトを観測することが難しい．そして，検知システムはフィッシングサイトの Web コ

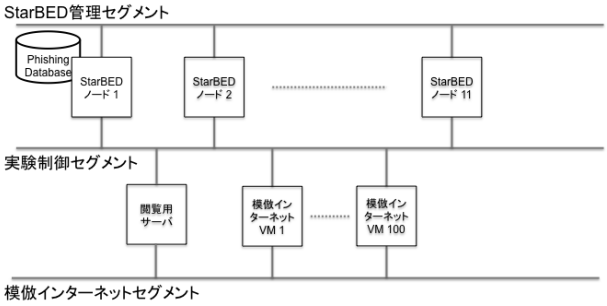


図 20: phishcage の実験構成

ンテンツだけでなく，IP アドレスや FQDN についての情報，例えば AS 番号や whois 登録情報なども確認しているが，これらの情報も日々刻々と変化しており，こうした情報の保存も必要である．さらに，フィッシングサイトはボットネットに発見されることも多く，ボットネット検知技術や他のインシデントとの連関分析などを用いたフィッシングサイトの検知システムも必要であるが，この場合，ネットワークトポロジの再現も必要となる．

そこで，フィッシングサイトを模倣インターネット上に再現する phishcage の研究開発に取り組んだ．模倣インターネットは実インターネットと隔離された空間に設置する必要があるが，フィッシングサイト検知システムが用いる情報，例えばグローバル IP アドレスや whois といった情報も再現する必要がある．

2011 年 11 月に行われた MENS 2011 の実験では，フィッシングサイトの再現をまず実施することとした．図 20 に概要を示す．フィッシングサイトのコンテンツを保存するサーバを用意し，模倣インターネットを構成するノードの管理用セグメントから接続し，コンテンツの取得を行う．実験用のセグメントには AnyBed の機能を用いて実インターネットの IP アドレスを割り当てる．また，模倣インターネットに DNS サーバを設定し，FQDN と IP アドレスの変換を行えるよう実施した．

次に，環境作成の手順を示す．

1. StarBED ノード 10 台にフィッシングサイト再現用の VM インスタンスを合わせて 100 個分稼働させる．
2. StarBED ノード 1 台にフィッシングサイト閲覧用の VM インスタンスを 1 個分稼働させる．
3. 保存してあるフィッシングサイトのデータベースを，StarBED ノード 1 台に移植する．
4. AnyBed により，日本国内においてピア数の多い上位 100 AS のトポロジを作成する．
5. フィッシングサイトが発見された AS が，国内 TOP 100 に含まれ知恵るか観測する．
6. データベースには存在しているが，TOP 100 ではない AS の番号をピックアップする．今回の実験では 5 つの AS が該当した．
7. AnyBed により，日本国内の上位 95AS と，フィッシングサイトのある 5AS の合計 100AS をリスト化し，このリストを基にトポロジを生成する．
8. 作成したトポロジに含まれている AS の個数が 100 個であるかを調べる．今回の実験では離れ小島問題もあり 99 個の AS しか発見できなかった．そこで，含まれなかった AS のリンクを持つ 1AS を加え，先ほどのフィッシングサイトが発見された 5AS と日本国内の上位 94AS をリスト化，トポロジを作成することで 100AS の環境を実現した．
9. フィッシングサイトの FQDN に対して DNS 正引きを行うと，実インターネットにおけるフィッシングサイトが発見された AS と，同じ番号の AS に誘導されるよう DNS レコードを設定する．

今回の実験で用いたフィッシングサイトのデータベースはフィッシングサイトの URL に関するデータベースと，コンテンツに関するデータベースがある．URL データベースには，識別番号，IP アドレス，AS 番号，FQDN，URL が含まれており，別のデータベースには識別番号（URL データベースと同じもの）と，コンテンツが含まれている．フィッシングサイトの URL にアクセスすると，DNS によってフィッシングサイトの AS の持つ IP アドレスが返される．Web ブラウザがコンテンツを取得するため当該サーバにアクセスすると，当該サーバはアクセス元の URL からフィッシングサイトの識別番号を取得し，この識別番号を基にコンテンツをデータベースより取得，Web ブラウザに応答する．なお，Web コンテンツから他のウェブコンテンツを呼び出している場合は，そのコンテンツも再生されるよう同一セッションの間では同じ識別番号が

使われるよう管理していたり，あるいは呼び出し元の URL から識別番号を類推したりといった作業を，コンテンツ再現サーバが実施している．

今回の実験では，上記手順 (2) で作成した確認用の VM インスタンスに Windows 7 をインストールし，Internet Explorer による確認を行った．概ね正しくフィッシングサイトを再現していたが，2 つのコンテンツ再現がうまくいかない事例を発見した．1 つめは，フィッシングサイトが SSL を利用した暗号化通信を利用している場合である．フィッシングサイトのコンテンツを Proxy サーバで取得しているために起こった問題であり，コンテンツの取得方法を考慮することによって解決したい．また，2 つめは DNS レコードを使わず IP アドレスを直接 URL に埋め込むことによってアクセスしている場合の例である．これは，模倣インターネットのサーバに実際のフィッシングサイトと同じ IP アドレスをエイリアス IP として割り当てることによって解決したい．

5 イベント

Nerdbox Freaks および DeepSpaceOne WG のイベントとして，昨年度に引き続き MENS (Multiple Experiments for Nerdbox/Nebula on StarBED) と称した連携実験を開催した．また，9 月 WIDE 合宿にて Fun with StarBED Cubic と題し，参加者を募って 9 月 WIDE 合宿前の 1ヶ月間の間に実験を実施してもらい，9 月合宿にて実験結果とライブデモンストレーションを実施するという新たな試みを行った．また，開発ワークショップとして今年度は Router Hackathon を実施した．

学会や展示会などで開催されたコンペティションに参加し，模倣環境構築のデモンストレーションを実施した．また，クラウドコンピューティングデベロッパーズワークショップと題し，開発者向けの合宿形式のハンズオンワークショップを開催し，開発合宿を通じての技術交流を実施した．

5.1 MENS

StarBED 上での模倣環境に関する実験の連携実施と，実験技術の共有を目的として，情報通信研究機構 北陸 StarBED 技術センター (StarBED) にて，MENS2011Spring(Multiple Experiments for Nerdbox/Nebula on StarBED 2011 Spring) および MENS2011Autumn を昨年から引き続き開催した．Nerdbox-freaks および DeepSpace1 WG の有志が参加した．

5.1.1 MENS2011Spring

MEMS2011Spring は 4 月 18 日から 6 月 30 日までの期間で実施された．今回はこれまでに StarBED を利用した経験がある参加者が多く，北陸 StarBED 技術センターに集合して作業をおこなうという形態を取らず，参加者がそれぞれ VPN を利用してリモートから StarBED 環境に接続して実験を行った．それに応じて利用期間も比較的長くとることが可能となり，十分な実験が行われた．

参加した実験プロジェクトは，大規模実験環境への OS イメージ投入の高速化，テストベッド間協調，uKOI (universal Keying Over the Internet)，大規模なネットワーク網を利用したトラフィック生成，仮想ノードを用いた BGP 網エミュレーション，Traceback などである．また，MENS2011Spring で実施された実験は，各実験プロジェクトによって様々な国際学会や WIDE 研究会といった場で発表された．

5.1.2 MENS2011Autumn

MENS2011Autumn は，11 月 1 日から 12 月 2 日からのほぼ 1ヶ月の期間で開催され，今回も遠隔からの利用者が中心となった．

前回に引き続き，テストベッド間協調，仮想ノードを用いた BGP 網エミュレーション，Traceback などのプロジェクトが参加したほか，C++用並列分散処理ライブラリの評価，大規模災害発生時の IMS インフラにおける障害回避手法，フィッシングサイトの模倣などの実験が新しく実施された．

5.2 Fun with StarBED Cubic

2011 年 9 月 WIDE 合宿の実験として「Fun with StarBED Cubic」と題した実験を Nerdbox Freaks にて実施した．この実験では，あらかじめ StarBED を利

図 21: 第 1 回 Router Hacakthon の様子

用した大規模実験参加者を募り，9 月合宿開催の 1ヶ月前となる 8 月上旬から実験を実施し，実験した結果を 9 月 WIDE 合宿の研究発表枠にて発表してもらう，研究発表連動型実験として開催した．また，公募した実験参加者以外に，合宿ネットワークの一部として camp PC も利用し，WIDE Cloud 環境や NAT64 ゲートウェイ，syslog サーバなどを構築した．

詳細に関しては第 X 部「大規模な仮設ネットワークテストベッドの設計・構築とその運用」に記載し，ここでは割愛する．

5.3 Router Hackathon

昨年までの CCDW (Cloud Computing Developers Workshop) や SNDW (StarBED Nebula Developers Workshop) を踏襲し，今年度はオープンソースルータに代表されるオープンソース開発や開発者間の技術交流を目的として Router Hackathon と題した開発ワークショップを Nerdbox Freaks で実施した．第 1 回 Router Hackathon は共愛学園前橋国際大学 小柏伸夫准教授の協力のもと，10 月 15 日から 10 月 18 日までの 4 日間，共愛学園前橋国際大学にて実施した．続いて，第 2 回 Router Hackathon は名古屋大学 河口信夫教授の協力のもと，12 月 16 日から 12 月 19 日の 4 日間，名古屋大学にて開催した．図 21 は第 1 回 Router Hackathon の様子である．

Router Hacakthon では何かしらの成果物を期間中に仕上げることに重きを置いた合宿形式の開発ワークショップである．Router Hackathon では期間中であればどの日程でも参加可能である．ただし，参加者には参加初日に期間中の開発目標を宣言し，各参加者の参加最終日に達成度と満足度を自己評価する事で密度の高い開発を実施する．また，昼食，夕食前に開発の進捗を確認し，開発を進める形式で行っている．開発物はソフトウェアルータのみに限定しておらず，オープンソースソフトウェアであれば特に問わない．また，期間中に論文やマニュアルなどの文章執筆を実施し，論文やマニュアルなどを成果物としてもよい．

第 1 回参加者と各自の成果物，達成度，満足度は以下のとおりである．

- 小柏伸夫 （共愛学園前橋国際大学）
  - **参加期間：**10 月 15 日〜10 月 18 日
  - **開発物：** ogashiwan 2011
  - **開発物概要：** python を利用した XMPP アプリケーション開発フレームワークを実装した．
  - **達成度：** 100 %
  - **満足度：** 100 %
- 上野幸杜（慶應義塾大学）
  - **参加期間** 10 月 15 日〜10 月 18 日
  - **開発物** LISP XTR and Map server software implementation
  - **開発物概要** IETF で標準化が進んでいる LISP (Location ID Separation Protocol) における XTR (LISP Tunnel Router) および Map サーバのソフトウェア実装を実装した．
  - **達成度** 100 %
  - **満足度** 80 %
- 中村遼（慶應義塾大学）
  - **参加期間** 10 月 15 日〜10 月 18 日
  - **開発物** Overlaid IP multicast Router
  - **開発物概要** P2P オーバーレイによって仮想的にレイヤ 2，レイヤ 3 ネットワークを構築し IP マルチキャストルーティングを行うソフトウェアルータを実装し，StarBED を使った実験環境の構築を実施した．

  - **達成度** 65 %
  - **満足度** 85 %

- 遠峰隆史（慶應義塾大学）
  - **参加期間** 10 月 15 日〜10 月 18 日
  - **開発物** F2R (Fuzzing test Framework for Routers)
  - **開発物概要** FreeBSD JAIL を利用し，injector と receiver の間に（ソフトウェア）ルータを挟み込み，StarBED のような大規模実験施設を利用して並列に Fuzzing テストを行うルータ用検証フレームワーク F2R のプロトタイプを実装した．
  - **達成度** 75 %
  - **満足度** 85 %

- 岡田行央（ソフトバンクテレコム）
  - **参加期間** 10 月 15 日〜10 月 18 日
  - **開発物** HagaUme の論文
  - **開発物概要** 9 月 WIDE 合宿は MENS2011 で実験した HagaUme（P2P オーバレイネットワークを用いたサーバ死活監視・サービス品質監視フレームワーク）の論文執筆を 2 本仕上げることを目標とし，最終的には論文を 1 本完成した．
  - **達成度** 50 %
  - **満足度** 80 %

- 関谷勇司（東京大学）
  - **参加期間** 10 月 15 日〜10 月 18 日
  - **開発物** AirDB
  - **開発物概要** IRR データベースの登録情報から CISCO, Juniper, GNU Zebra (Quagga) などの各種ルータ用の経路フィルタを生成するツールを開発した
  - **達成度** 80 %
  - **満足度** 150 %

- 櫨山寛章（奈良先端科学技術大学院大学）
  - **参加期間** 10 月 15 日〜10 月 18 日
  - **開発物** StarBED 利用 TIPS の整理
  - **開発物概要** 中村，遠峰の StarBED 利用の補佐を実施しながら，StarBED 利用における TIPS を整理し，StarBED 技術センターの運用者らにフィードバックした．また，遠峰の実験補佐として、StarBED 上に i386，amd64 それぞれのアーキテクチャにて FreeBSD JAIL のディスクイメージを作成し pickup によるディスクイメージの吸い出しを実施した．
  - **達成度** 70 %
  - **満足度** 70 %

- その他の参加者として，前橋国際大学の学生や卒業生らにローカルアレンジとしてインターネット接続環境の整備や買い出しなどを実施していただいた．また，前橋国際大学小柏研究室のゼミナールの一環として数名の学生が Router Hackathon の様子を見学した．

第 2 回参加者と各自の成果物，達成度，満足度は以下のとおりである．

- 岡田行央（ソフトバンクテレコム）
  - **参加期間** 12 月 16 日〜12 月 19 日
  - **開発物** HagaUme の論文
  - **開発物概要** 第 1 回 Router Hackathon でやり残した HagaUme に関する 2 本目の論文を執筆した．
  - **達成度** 80 %
  - **満足度** 80 %

- 関谷勇司（東京大学）
  - **参加期間** 12 月 17 日〜12 月 19 日
  - **開発物** AirDB, 2012 年 1 月 IA 研究会投稿論文
  - **開発物概要** EBGP エミュレーション実験環境構築ツールである AnyBed と AirDB による経路フィルタ生成機能を連携するために，櫨山と協力し実装のすり合わせを実施した．また，2012 年 1 月に JANOG 共催イベントとして開催される IA 研究会への投稿論文を執筆した．

- **達成度** 80 %
- **満足度** 80 %

- 石原知洋（東京大学）
  - **参加期間** 12 月 16 日〜12 月 19 日
  - **開発物** createzones, Debian FAI for StarBED
  - **開発物概要** puppet, bind9, unbound を用いた DNSSEC エミュレーション環境構築ツール createzones を AnyBed による EBGP エミュレーションと連携するために，櫨山と協力して実装間のすり合わせを実施し，StarBED 上に EBGP エミュレーションと DNSSEC エミュレーションの結合環境を構築した．また，奈良先端大 榎本と連携し Debian 6 の FAI を StarBED 向けにカスタマイズし，PXEBoot から FAI を実行することに成功した．借用ノードの間違いや i386, amd64 のイメージを間違える，bswc.pl の不具合により VLAN 設定が意図したものにならないなど，ベースとなる KVM 環境の構築に 2 日半費やしてしまったため，当初予定していた createzones と AnyBed の結合は自動連携するまでには至らなかった．
  - **達成度** 25 %
  - **満足度** 60 %

- 櫨山寛章（奈良先端科学技術大学院大学）
  - **参加期間** 12 月 16 日〜12 月 19 日
  - **開発物** AnyBed, pickup / wipeout の検証
  - **開発物概要** 関谷の AirDB, 石原の createzones と AnyBed の連携のため，AnyBed の拡張や AirDB の拡張を実施した．また，第 1 回 Router Hackathon で構築した FreeBSD JAIL イメージが pickup で正しくイメージの吸い出しが行えているか否かの検証を実施した．FreeBSD JAIL イメージは i386 のみディスクイメージの吸い出しが成功しており，amd64 イメージは残念ながらディスクイメージ吸い出しに失敗していたことが wipeout の実行から明らかとなった．石原と同じく，環境構築に 2 日半費やしたため達成度は低い結果となった．
  - **達成度** 25 %
  - **満足度** 60 %

- 榎本真俊（奈良先端科学技術大学院大学）
  - **参加期間** 12 月 16 日〜12 月 19 日
  - **開発物** fedd extention for StarBED, Debian FAI for StarBED
  - **開発物概要** USC-ISI Deterlab と共同開発を行っている異種エミュレーションテストベッド連携技術の研究開発の一環として，USC-ISI Deterlab が開発した Emulab Tool ベーステストベッド連携ツールである fedd に対し StarBED 向けの機能の実装に取り組んだイ．成果としては StarBED のリソース管理デーモンである ERM から予約資源情報の取得部分までは実装を完了した．また，石原と連携し，Debian 6 の FAI を StarBED 向けにカスタマイズし，PXEBoot から FAI を実行することに成功した．当初の予定では fedd の StarBED 向け拡張を完成する予定であったが，ERM のプロトコル，ProtoGENI RSpec の習得に予想以上に時間を費やす結果となり，結果として低い達成度となった．
  - **達成度** 25 %
  - **満足度** 60 %

- 岡田和也（奈良先端科学技術大学院大学）
  - **参加期間** 12 月 16 日〜12 月 19 日
  - **開発物** droppy
  - **開発物概要** Interop Tokyo NOC Team で利用している仮想ルータ環境や EtherOAM 環境における疎通性検証ツール droppy の改修を実施した．改修項目として，JAIL 環境の一括設定，ログデータベースの再設計と再実装，WIDE Cloud 向け EtherOAM 拡張を実施した．改修の結果，基本的な再設計は実施できたが，JAIL での traceroute とログデータベースの再実装は未完成のまま Router Hackathon の期間を終了した．

  - **達成度** 60 %
  - **満足度** 80 %

- 藤原礼征（ソリューションクルー）
  - **参加期間** 12 月 18 日
  - **開発物** Spring OS API の設計
  - **開発物概要** 昨年度まで CCDW を共同で開催していた株式会社ソリューションクルー藤原礼征氏を招待し，12 月 18 日の 1 日だけ Router Hackathon に参加していただいた．藤原氏は Spring OS API (プロジェクト名 eggrole) を StarBED 技術センターと共同開発することになり，StarBED を利用している Router Hackathon 参加者からヒアリングを行いつつ，eggrole のクラス設計を実施した．1 日だけの参加であったため，目標としてはプロジェクトの開始とヒアリングのみに設定していたため，達成度および満足度はともに 100%となった．
  - **達成度** 100 %
  - **満足度** 100 %

- その他の参加者として，名古屋大学の河口信夫教授や河口研究室の学生らにローカルアレンジとしてインターネット接続環境の整備や買い出しなどを実施していただいた．また，初日には河口研究室の学生も数名 Router Hackathon に参加し，河口研究室内の LISP, IPv6, NAT64, DNS64 環境構築や SAMTK 3D monitor クライアントの実装などを実施した．

Router Hackathon は 2012 年にも引き続き開催する予定で，開催時期は未定であるが東京大学 石原知洋助教の協力により東京大学駒場 1 キャンパスにて開催する予定である．また，第 1 回 Roueter Hackathon 参加者である共愛学園前橋国際大学 小柏伸夫准教授とともに 2012 年 3 月 WIDE 合宿のホットステージと連動した Router Hackaton である Open Hotstage を企画している．

6 おわりに

本年度の主な成果として，Deep Space One WG および Nerdbox Freaks WG と合同で大規模実験ワークショップや開発ワークショップによる新規実験者の開拓，WIDE 合宿との連携した大規模実験環境の利用，現行のテストベッド管理ツールや実験補助ツールの利用を促進，次期テストベッド管理ツールや実験補助ツールのグランドデザインの実施が挙げられる．

来年度も Deep Space One WG と Nerdbox Freaks WG との協調により，柔軟な実験環境を構築するためのツール群の研究開発とともに，ナレッジベースの整備および，より現実的な実験を行うための実験環境構築手法の研究開発も平行して行っていく．また，LACE (Long range Access to Collaborative Environments) Project において米国 Deterlab とのテストベッド連携実験およびテストベッド連携環境でのエミュレーション実験を実施する予定である．

参考文献

[1] 宮地利幸, 中田潤也, 知念賢一, ラズバン・ベウラン, 三輪信介, 岡田崇, 三角真, 宇多仁, 芳炭将, 丹康雄, 中川晋一, 篠田陽一, *“StarBED:* **大規模ネットワーク実証環境***”*, 情報処理, 第 49 巻, 第 1 号, pp. 57-70, 2008 年, 1 月.

[2] CAIDA Project, *“CAIDA Project, AS ranking.”* `http://as-rank.caida.org/`.

[3] CAIDA Project, *“Routeviews Prefix to AS mappings Dataset (pfx2as)”* `http://www.caida.org/data/routing/routeviews-prefix2as.xml`.

[4] Mio Suzuki, *“AnyBed: a testbed-independent topology configuration tool.”* `http://sourceforge.net/projects/anybed/`

[5] Shinsuke Miwa, *“XENebula.”* `http://tbn.starbed.org/XENebula/`

[6] StarBED Project, *“SpringOS version 1.5.”* `http://www.starbed.org/download/index.html`

[7] FAI Project, *"FAI - Fully Automatic Installation"*
http://fai-project.org/.

[8] Kenjiro Taura, *"FAI"*
http://www.logos.t.u-tokyo.ac.jp/~tau/mediawiki/index.php/FAI.