2012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)　田部英樹 (hideki-t@jaist.ac.jp)
村上正太郎 (shotaro@jaist.ac.jp)　Razvan Beuran (razvan@nict.go.jp)
三輪信介 (danna@nict.go.jp)

2012 年 12 月 15 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 Freks 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．

2011 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る．

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る．

Deep Space One WG 及び Nerdbox Freaks WG では，現在，主に StarBED を対象にして研究活動を行っている．StarBED および StarBED での実験補助ツールである SpringOS に関しては文献 [1] を参照していただきたい．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．

2 SpringOS の API 実装

StarBED [1] は情報通信研究機構 北陸 StarBED 技術センターに設置された 1000 台以上の実験専用の PC ノードを持つ大規模なネットワーク実験設備である．StarBED のような大規模なネットワークテストベッドにおいては，利用者の実験実行を支援するためにさまざまな支援用ソフトウェアが用意されている．SpringOS [1] は，主に StarBED での実験実行を支援するために我々が開発を進めてきたソフトウェアスイーツであり，主にテストベッドの実験用リソースの管理，実験ノードへのソフトウェア導入，スイッチ設定によるトポロジの構築，シナリオの実行などの機能を持っている．

図 1 に SpringOS のモジュール群を示す．SpringOS では，ある利用者に割り当てられているリソースを他の利用者が操作できないよう，テストベッド全体を管轄するモジュールが存在しており，これらが利用者間調停をおこなう．これに対して，それぞれの利用者が利用する管理ノードで動作する各種ユーザインターフェースに該当するモジュールが用意されており，利用者はこれらに必要な情報を入力することで，実験をおこなうためのリソースである実験用ノードやスイッチを操作する．これらのユーザインターフェースに対する入力は，SpringOS 独自のものであり，それぞれのモジュールごとに異なった形式での記述が必要である．

ネットワークテストベッドの利用者が実験をおこなう目的は，対象とするネットワーク技術の挙動を把握することであり，実験をおこなうためのツールを利用するための技術を習得することは本質ではないため，実験実行のためのツールはできるだけ簡単に利用できることが望ましい．また，これまでに SpringOS を利用してきた利用者からも，既存のプログラム言語や，ネットワークシミュレータの設定記述を利用したいという要望があった（図 2）．しかし，親しみのあるプログラム言語やネットワークシミュレータは利用者に

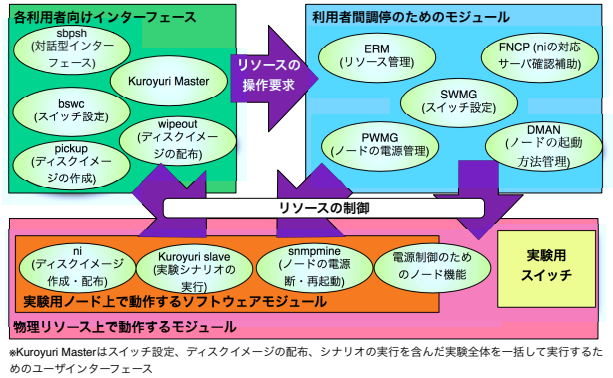

図 1: SpringOS のモジュール群

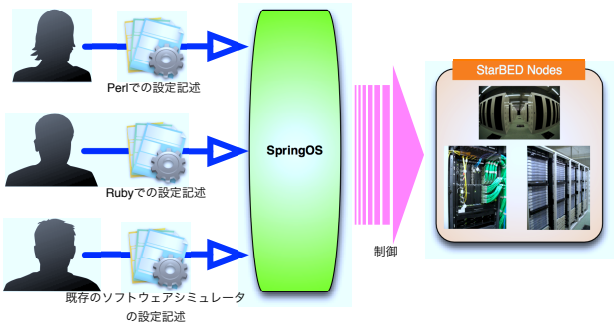

図 2: ネットワークテストベッド利用者からの要望

依存し，利用者の要望全てを StarBED/SpringOS で対応することは事実上不可能である．

これらの要求に応えるため，一般的なプログラム言語で SpringOS の各種機能を利用できるようなライブラリを開発した．本論文では，本ライブラリについて議論する．

2.1 既存の SpringOS ユーザインターフェース

図 1 に示したとおり，SpringOS は複数のモジュールで構成されるソフトウェアスイーツであり，利用者はそのうちユーザインターフェースとなるモジュールを利用してその挙動を制御し，実験を遂行する．これらのユーザインターフェースを利用して実行できる機能を表 1 にまとめた．

表 1: SpringOS のインターフェースとその機能

モジュール	機能
Kuroyuri Master	環境構築とシナリオ実行を含む実験の一括実行
pickup	ディスクイメージの作成
wipeout	複数のノードへのディスクイメージの配布
bswc	スイッチの VLAN 設定
sbpsh	各種 SpringOS の機能を提供するシェル

SpringOS は，複数の実験ノードに共通な OS を含むソフトウェア部分の導入のために，一台のひな形となる実験ノードからバイナリのディスクイメージを取得し，これを複数のノードに配布する手法を採用している．このためのひな形となるディスクイメージを作成するためのユーザインターフェースが `pickup`，配布するためのものが `wipeout` である．`Kuroyuri Master` は基本的にその他のインターフェースモジュールの全ての機能を持っており，実験を一括しておこなうためのものである [2]．これ以外のモジュールは `Kuroyuri Master` の機能の一部だけを利用するために用意されている．

`Kuroyuri Master` と `bswc` は，設定ファイルから入力された内容をバッチ処理するため，ツールを使うための設定および，それらを利用した実験リソースに対する処理の双方が一つのファイルに埋め込まれている．`pickup`/`wipeout` についても同様にバッチ的に処理がおこなわれるが，実行ごとに対象のノードなどを入れ替えながら設定ファイル自体は再利用するため実験リソースに対する処理はコマンドプロンプトのオプションとして渡されることが多い．ただし，これらの情報に関しても設定ファイルに記述することが可能である．また，`sbpsh` は対話型シェルであるため，実験リソースに対する処理に関する入力は `sbpsh` 起動後に受け付ける．

`Kuroyuri Master` は実験ノードのソフトウェア設定および実験ノード上でのコマンド実行によるシナリオ遂行の機能を持つため，書き込むディスクイメージ URI やパーティション情報，実行するコマンドや同期のためのメッセージ交換のための記述が含まれている．これに比較し `bswc` の設定ファイルには一行が一つのスイッチポートに対する命令が記述されている．ここでは，スイッチ上で実験ノードの NIC に接続されているポートの有効化をした後，それぞれの NIC に接続されたスイッチのポートを指定された VLAN に設定している．`sbpsh` は実験ノードが利用する OS がインストールされているパーティション番号を指定し再起動時にその OS が起動するように設定したり，ノードの電源の投入などがおこなえ，SpringOS のさまざまなモジュールとの通信が必要になるため，それぞれのサービスが提供されているノードのアドレス情報などが設定として必要になる．これらの設定中では，同じ情報がそれぞれの設定ファイルに別々の書式で記述されており，利用者は利用するモジュールごとにそれぞれの書式で同じ内容を記述するという冗長な作業が必要である．また，利用者間調停のためのモジュール群に接続するための IP アドレスやポートなどはユーザごとに変化しないため，それぞれの利用者から隠蔽することも可能である．これらの指定を誤っていることによるトラブルもこれまで散見されている．

新しいユーザインターフェースを実装することにより，この問題は解決できるが，SpringOS の各モジュールは複雑な相互協調の上で動作しており，全てを理解した上で実装をおこなうことは，実験の結果を得ることが本質的な目的である利用者にとっては大きな負担となる．また，実験環境を構築するためのノードへの OS の導入や，スイッチ設定といった一連の作業をおこなう際に利用者が親しみ深い Perl や Ruby といった一般的なスクリプト言語やネットワークシミュレータの設定ファイルを利用したいという要望が上がってきている．

以上をまとめると，現状の問題および要望は以下である．

1. SpringOS 各ユーザインターフェースの設定記述方式が同一でない
2. SpringOS 各ユーザインターフェースの出力の理解が困難である
3. 新しいユーザインターフェースを実装する負担が大きい
4. より一般的に利用されている言語での設定記述を

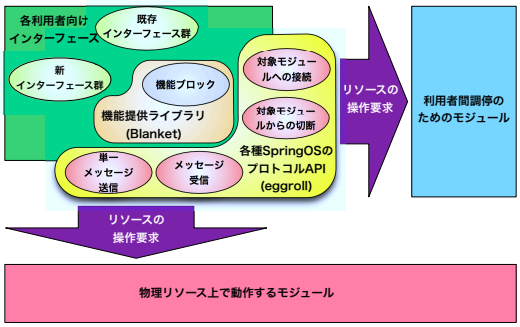

図 3: SpringOS の API 境界面

おこないたい

2.2 SpringOS API の提案と実装

前節であげたように現在の SpringOS のユーザインターフェースは全てのユーザにとって利用しやすいものではない．しかし，利用者の習得している言語やシミュレータの種類，そして StarBED および SpringOS の理解の深さによってその使い勝手は一定では無いため，すべての利用者が使いやすいユーザインターフェースを構築することは事実上不可能である．このため，さまざまな要求を持ち技術レベルが異なる利用者にそれぞれが使いやすいと考えるユーザインターフェースを作成して貰い，これを利用者間で共有する場を提供することを考えた．このため，SpringOS の複雑な処理手順を隠蔽した上で，SpringOS の機能群をさまざまな言語で利用できるようにすることで，利用者による実装を容易にするための API を用意するアプローチを取った．API を利用して SpringOS の機能を用いることにより，一般的に利用されている言語での設定記述が可能となる．さらにこれを設定記述として利用するのでは無く，新しいユーザインターフェースを実装するために利用する事で，新しいユーザインターフェースを実装するための負担を低減できる．結果的にこれは新しいユーザインターフェースの実装を促進することになり，設定記述の方式および出力をよりわかりやすいものに整理することに繋がる．

2.2.1 API 設計

図 3 に今回提案する API モジュールの境界面を示す．今回の API では，基本的には，SpringOS の既存ユーザインターフェースモジュールが利用している各種プロトコルでのサーバに相当するモジュールとのセッションの確立および切断，メッセージの送信およびサーバモジュールからのメッセージの待ち受けの 4 種類に相当する関数をそれぞれのサーバモジュールごとに用意する事とした．しかし，これだけではプロトコルの詳細を理解する必要があり，利用者が使いやすいとは言えないため，これらの関数を利用して，SpringOS のそれぞれの機能を容易に利用できるライブラリ群を用意する事とした．前者を eggroll と呼び，eggroll を利用して実装した後者のライブラリを Blanket と呼ぶ．eggroll と Blanket の関係をネットワークテストベッドに存在している実験ノードの割り当てを受けたい場合を例にして紹介する．本手順には，以下の手順が必要である．

1. リソースマネージャにログイン
2. 利用可能なノードのうち必要な機能を満足するノードがあるかを検索
3. ノードがあればそのノードの割り当てを要求
4. リソースマネージャからログアウト

これらの手順を eggroll が提供し，ノードの必要条件を渡せば一括してこれらの処理をおこなう関数を Blanket が提供する．

これにより現状の SpringOS の各ユーザインターフェースは，これまでのまま直接 SpringOS の各種モジュールと通信をおこない，今後開発がおこなわれるであろうユーザインターフェースは，1. 各種モジュールと直接通信をおこなう，2. eggroll が提供する API を利用する，3 Blanket が提供する機能ごとにまとめられた関数を利用する，という 3 種類の方法で SpringOS の機能を利用できるようになる．

2.2.2 API 実装

前節で述べた設計を実現するために C++で記述したライブラリを Simplified Wrapper and InterfaceGenerator（SWIG）[3] を利用してその他のスクリプト言

語から利用できるように eggroll の実装をおこなった．SWIG は Perl，PHP，Python，Tcl，Ruby といったスクリプト言語などに対応しており，利用者の慣れ親しんだ形での実験記述をある程度実現できると考えられる．また，これ以外の独自言語についても既存のスクリプト言語ではパーサなどを記述するためのライブラリが用意されていることも多いため，さらに多くの言語に対応できると考えられる．

今回実装の対象とした SpringOS の機能は以下の通りである．

- 実験リソースの管理と割り当てをおこなう ERM との通信
- スイッチ設定をおこなう SWMG との通信
- ノードの起動方法を管理する DMAN との通信
- ノードの死活管理をおこなう PWMG との通信
- ディスクイメージを作成するために利用される ni, FNCP との通信
- 一括した実験遂行をおこなう Kuroyuri Master からの各実験ノードの状態を確認するための通信

これらを利用すれば，表 1 に挙げたモジュールのうち，`Kuroyuri Master` によるシナリオ実行に関するものを除いたモジュールの機能が実現可能である．

今回は C++で記述したプログラムコードを SWIG を利用して Perl から利用できるように実装をおこなった．また C からも利用できるようにも実装している．

2.3 SpringOS API を利用したユーザインターフェース実装例

eggroll を利用して，ディスクイメージの作成と書き込みをおこなう `pickup` と `wipeout` 相当の機能を持つ新しいインターフェースをモジュールを作成した．

SpringOS を利用したディスクイメージの作成は以下の手順でおこなわれる．

1. 対象のノード（群）をその利用者が予約しているかを ERM に確認し，予約が確認できれば割り当て
2. ディスクイメージの読み込みもしくは書き込みをするためのモジュールである ni がインストールさ

```
<?xml version='1.0'?>
<facility>
  <groups>
    <group>
      <name>A</name>
      <niloader>recover_system/pxeboot</niloader>
      <nikernel>recover_system/kernel</nikernel>
      <nidisk>FreeBSD/ni-disk.fs</nidisk>
      <nidevice>/dev/rda0</nidevice>
    </group>
    <略>
  </groups>
  <services>
    <service>
      <name>ERM</name>
      <ipaddr>ERM の IP アドレス</ipaddr>
      <port>ERM のポート番号</port>
    </service>
    <service>
      <name>DMAN</name>
      <ipaddr>DMAN の IP アドレス</ipaddr>
      <port>DMAN のポート番号</port>
    </service>
    <略>
  </services>
</facility>
```

図 4: diskhandle.pl の施設情報設定

```
<?xml version='1.0'?>
<userconf>
  <user>
    <name>ERM のユーザネーム</name>
    <passwd>ERM のパスワード</passwd>
    <project>ERM のプロジェクト名</project>
  </user>
  <encd>
    <ipaddr><diskhandle.pl が動作するノードの IP アドレス></ipaddr>
    <port>diskhandle.pl が通信をおこなうポート番号</port>
  </encd>
  <diskimage>
    <target>A001:1</target>
    <fileserver>ftp://guest:guestpass@10.0.0.1/</fileserver>
    <serverdir>/diskimage_dir/</serverdir>
    <uploadfile></uploadfile>
    <downloadfile>a001-disk-partition1.gz</downloadfile>
  </diskimage>
</userconf>
```

図 5: diskhandle.pl のユーザごとの情報設定

れた OS でネットワークブートするよう，DMAN に起動方法の変更を指示

3. 起動してきた `ni` が通信をおこなうモジュール（`pickup` もしくは `wipeout`）の IP アドレスを FNCP に登録
4. PWMG に対象となるノード（群）の電源投入もしくは再起動を指示
5. 対象ノードの起動とともに起動してきた ni に対して書き込みもしくは読み込みをおこなうデバイス名と保存するためのファイルサーバの URI を指示
6. 処理終了後起動方法を変更し再起動
7. ERM に対象ノード（群）の解放を指示

これらの手順により今回実装をおこなった SWMG との通信および `Kuroyuri Master` との通信による実験ノードの状態保持以外の検証が実践的に可能である．

実装手順としては，eggroll を利用して，ERM とのノードの確保および解放のための手続きや，DMAN への起動方法の変更指示といった単位ごとに Perl で Blanket 相当のライブラリを用意し，それを利用したユーザインターフェース `diskhandle.pl` を実装した．

設定は，今後実装する別のユーザインターフェースと共有できるよう，SpringOS が利用する情報を網羅するように設計をおこない，また，一般的な利用者が変更をおこなう必要がない設定と各ユーザが変更する必要がある設定を記述するファイルを分離した．このファイルはさまざまな言語でパーサが提供されているという点から XML を利用して記述することとした．図 4 と図 5 にそれぞれの設定ファイルを示す．基本的に利用者が知る必要が無い情報には，ni が起動するためのディスクレスシステムのカーネルやディスクイメージの情報，さらにこれらが認識するディスクのデバイス名，そして ERM や DMAN へ接続するための情報が含まれる．一方，利用者が設定する必要がある情報としては，各ユーザごとに割り当てられる ERM のアカウント情報，`diskhandle.pl` を動作させるノードの IP アドレスや，ディスクの読み書きをおこなう対象となるノードの情報が記述される．ただし，これらの情報はコマンドラインからも上書きして設定できるように実装した．

Blanket 部分は今回新たに実装したが，他の利用者とも共用ができるものである．したがって今後利用者は，これらの関数を呼び出す部分だけ実装すれば良い．今回実装した `diskhandle.pl` はエラー処理やヘルプなど含めて 300 行以下で記述可能であった．

詳細については、wide-paper-deepspace1-myadicomo2012-00.txt を参照していただきたい．

3 SummerOS

Open Flow などの新しいレイヤ 2 ネットワーク構築技術やより大規模な実験をより省力化して実験するために，StarBED での検証環境構築支援システムである SpringOS に対して拡張が必要になっている．そこで，deepspace-one WG では SpringOS 後継である SummerOS の研究開発に取り組み始めた．

SummerOS では，実験環境の構築における構成要素をまとめて扱うシナリオ拡張である C(Complex)-extension，ネットワーク構築に関する機能拡張である T(Topology)-Extension，万単位の仮想ノードを取り扱うための拡張機能である M(Massive)-Extention の 3 種類の機能拡張をおこなう予定である．以降，それぞれの機能拡張について，簡単に説明する．

3.1 C-Extension

C-Extension は，実験環境の各構成要素を複合体として扱うことで，より容易な実証実験の環境構築を目的とする拡張である．ネットワーク実験ではノードとネットワークの組み合わせから実験環境は構築されている．現状では，すべての要素を記述しなければならない．そこで複数のノードとネットワークの 構成要素を複合体として考案し，これをモジュールと呼ぶ．モジュールにパラメータを与えることで，実験者の目的に沿ったモジュールとして要素の変更が可能である．目的に合わせてモジュールを組み合わせることで，より少ない記述で実験・検証の環境構築を行う準備が可能となる．

3.2 T-Extension

StarBED 及び SpringOS は，現状では実験ネットワークの構築に VLAN（802.1Q）を用いることを想定

している．しかし VLAN は作成する 論理ネットワークの識別子である VID の数が 12bit と仕様で定められており，それを超える数の論理ネットワークを作成することはできない．すなわち，構築できる実験ネットワークの規模はその実験設備が作成できる論理ネットワークの数に依存する．また，StarBED では実験リソースを同時期に実験を行なっている他の実験者と共有する運用形態をとっている為，実験リソースの一つであるネットワーク識別子の数は同時に収容できる実験者の人数にも影響を与える．

T-Extension では，このような実験ネットワークの構築に関する現状の課題を解決することを目的とし，次世代のネットワーク制御技術 を採用した実証実験環境の設計・開発を行なっており，実験者に対して自由度の高い実験ネットワーク構築を提供する枠組みを検討している．

3.3 M-Extention

SpringOS に含まれているシナリオコントローラである kuroyuri は，現状では単純な master-slave モデルで設計されている．そのため，kuroyuri で扱う実験ノード数が万単位を越える場合には，設定や実験ノードの状態管理が困難になる恐れがある．そこで，kuroyuri に階層的モデルおよび多重化を導入する，M-Extention を設計・試作した．M-Extention では，物理ホスト上にプロセスや仮想マシンを用いてノード向けシナリオ評価器 (kusa) を多数起動する．M-Extention の試作ではプロセス多重化による kusa の多数起動を実現した．加えて各シナリオ評価器の通信を単一の TCP 通信に多重化し，全体評価器 (kuma) へ中継することで TCP ポート番号の消費を抑えている．今後はさらなる性能向上のための M-Extention のチューニングを行う予定である．現状の M-Extention では 200-300 台規模の動作実績を持つため，階層化後には万単位の動作が期待できる．

4 テストベッドフェデレーション

StarBED と USC/ISI が運用する Deterlab で協調した実験環境の構築を可能とする補助ツールとして，Collaborative Testbed federation(CTF) アーキテクチャの設計および開発を行った．StarBED と Deterlab は実験環境の構築に用いるツールが異なり，CTF ではユーザに対してそれぞれのテストベッドで実験環境の構築に用いるツールの違いを意識することなく実験環境の構築を可能とするアーキテクチャを目指している．

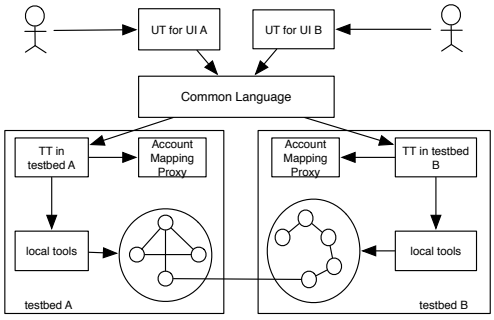

図 6: FTA architecture

4.1 CTF 設計

まずはじめに，CTF を設計するに当たり StarBED と Deterlab のテストベッドで実験環境構築手法およびテストベッドの運用のポリシの違いについてまとめる．StarBED と Detrelab の間では，構築方法については設定可能な資源のパラメータの違いが存在する．例として，Deterlab ではネットワークトポロジを設計する際にネットワークのリンクスピードを設定しなければならない．一方，StarBED ではリンクスピードの設定は行えず，常に一定の値となっている．また，Deterlab では OS は Deterlab であらかじめ用意しているイメージを指定する必要があり，ユーザは指定のイメージを改変することで実験に用いる環境を構築する．StarBED ではあらかじめ用意されているイメージを使うことも可能だが，ユーザには貸し出されたサーバに対して OS をインストールすることが許可されているため，ユーザが好みの OS をインストールし実験環境を構築することも可能である．最後に，Deterlab では実験ネットワークの分離はユーザが記述したネットワークトポロジをもとに自動的に設定される．StarBED では VLAN の ID 空間を切り分けてユーザに提供しており，ユーザは割り当てられた ID 空間を利用してネットワークの分離を行う．

次に，StarBED と Deterlab の運用ポリシの違いに

ついて述べる．一つ目の違いは資源の予約方法である．StarBED では，資源の予約は使用する資源と期間をあらかじめ決める．Deterlab ではユーザが実験を行う時に，テストベッドの資源に空きがありユーザの実験トポロジが構築可能な場合に資源をユーザに割り当てる．二つめの違いは，ユーザアカウントである．Deterlab ではユーザ ID とユーザパスワードで管理しており，ユーザは任意のプロジェクトに関連付けられてる．一方，StarBED ではユーザ ID，パスワードと プロジェクト ID で管理されている．しかし，ユーザ ID は各プロジェクトでひとつしか定義されておらず，すべての実験者が共通のユーザ ID，パスワードとプロジェクト ID を共有する．

よって，CTF では以下の５つの機能について設計を行った．一つ目は資源の管理機能，二つ目はユーザアカウントの管理機能，三つ目はユーザに貸し出された資源の制御を行う機能となる．また，ユーザにテストベッドの構築手法の差異を意識させずに環境構築を行うことを可能するために User side Translator(UT)，各テストベッドのシステムの構築形態と運用形態を変えることなくテストベッドフェデレーション可能にする Testbed side Federation(TT) の二種類の翻訳機を設計する．図 6 に CTF のアーキテクチャを示す．

4.2 CTF の実装

4.1 で述べた設計を元に，CTF の実装を行った．CTF は Emulab tools で管理されたテストベッドのフェデレーションを対象とした Deterlab Federation Architecture(DFA) とのフェデレーションを行うために，DFA を拡張する形で実装した．

今回のプロトタイプ実装では，二種類の翻訳機のうち UT は DFA ですでに実装されているものを利用し，TT の実装およびユーザアカウントの管理機能および資源の管理機能を実装した．本実装の検証として，CTF を用いて StarBED と Deterlab の資源にまたがる形で構築された実験環境上で２種類の検証実験を行った．一つ目は，ネットワーク機器に備え付けられた DNSSEC 用の鍵の自動更新手法に関する検証実験と２つ目は IRR (Internet Routing Registory) の登録情報に従い，ルータの経路設定を自動的に更新，調整するルーティング制御アーキテクチャに関する検証実験である．

本検証実験を行った結果，基礎的な環境の構築が可能であることが確認された．しかし，資源の管理方法については改善が見込まれるため，この点は今後の課題として継続して研究を行う予定である．

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，開発ワークショップである Router Hackathon を実施した．また，今年度は AsiaFI 2012 Summer School にて StarBED Hands-on Tutorial を実施し，AnyBed を用いた BGP エミュレーション実験と QOMET を用いた無線エミュレーション実験のハンズオンチュートリアルを実施した．

5.1 9 月 WIDE 合宿での StarBED を利用した合宿ネットワーク構築の省力化実験

9 月 WIDE 合宿の PC メンバと連携して，合宿ネットワークのルーティング機能やサーバ機能を StarBED 上に構築し，合宿地であるホテルには，EtherIP の終端装置，レイヤ 2 スイッチおよび無線 LAN 機器のみにして，合宿ネットワークの事前準備の拡充と現地での設営の省力化実験に取り組んだ．結果としては，前泊時の設営はかなり省力化でき，検証時間に時間を大きく割くことができた．詳細に関しては，2012 年 9 月 WIDE 合宿実験報告を参照していただきたい．

5.2 Router Hackathon

2012 年度も，2011 年度と同様に，研究開発ワークショップとして Router Hackathon を開催した．また，2012 年度は WIDE 合宿にて，ワークショップ形式での Router Hackathon として，2012 年 3 月合宿では Open Hotstage，2012 年 9 月合宿では WIDE Hackathon を開催した．Open Hotstage，および WIDE Hackathon の実施内容に関しては，WIDE 合宿の報告を参照していただきたい．

Nerdbox Freaks で開催した Hackathon の実施内容を下記に列挙する．

- **Router Hackathon 2012 Spring**

  2012 年 4 月 25 日（水）から 4 月 27 日（金）までの 3 日間，石川県能美市 StarBED 技術センターにて実施した．参加者は 7 名であった．合宿の成果として，DNSSEC / BGP Emulation 実験統合環境の試作，Fully Automated Install (FAI) による StarBED ノードへの OS インストールの実験，RADB 上に緊急避難 BGP パスを設定する拡張（AirDB）の試作，VXLAN の試作，SA46T-AT の試作を行った．このうち，SA46T-AT に関しては，handmade WG の報告に詳細を記載しているので，参照していただきたい．

- **Router Hackathon 2012 Summer**

  2012 年 7 月 25 日（水）から 7 月 27 日（金）までの 3 日間，静岡県熱海市 ホテルニュータカハシにて実施した．参加者は 5 名であった．合宿の成果として，仮想ネットワーク環境構築プラットフォーム（GINEW）の設計，ネットワークの抽象化方法，Web UI の設計，VRF，LSP 操作モジュールの試作を行った．

- **Router Hackathon 2012 September**

  2012 年 9 月 8 日（土）から 9 月 10 日（月）までの 3 日間，秋田県大仙市からまつ山荘にて実施した．参加者は 4 名であった．合宿の成果として，Deterlab と StarBED 間のテストベッドフェデレーション環境の構築事件，DNSSEC シミュレータの実験，RADB 経路フィルタ（AirDB）の vyatta 実装の試作を行った．

- **Router Hackathon 2012 October**

  2012 年 10 月 24 日（水）から 10 月 26 日（金）までの 3 日間，京都府京都市 大学コンソーシアム京都にて開催した．参加者は 7 名であった．合宿の成果として，Deterlab - StarBED 間に構築したテストベッドフェデレーション環境上での AirDB 実験環境の整備，GINEW フレームワークの文書化，JUNOScript を用いた GINEW モジュールの作成，ginewOS CLI モジュールの作成を行った．

Hackathon 参加者からは Hackathon 開催についておおむね好評である．2013 年度も不定期に Hackthon を開催する予定である．

5.3 AsiaFI Summer School StarBED Tutorial

2012 年 8 月に京都大学で開催された Asia Future Internet Forum (AsiaFI) の Summer School において，8 月 23 日を利用し，StarBED Hands-on Tutorial を実施した[1]．このハンズオンチュートリアルの目的は，想定参加者であるネットワーク分野の研究者にむけ， StarBED を利用した大規模実験を実際に利用しながら学習する事により，今後の研究へ役立ててもらう事であった．

午前は講義形式の発表を 3 件（図 7 参照），午後は「BGP エミュレーション」と「無線エミュレーション」の 2 つのテーマに基づきハンズオンチュートリアルを実施した．参加者は午前 30 名近く，午後は 11 名であった．午後の参加者のうち 3 名は中国からの参加者で，残りは日本の参加者であった．また，参加者のほとんどは今まで StarBED を利用した経験のない方で，個人で構築した実験環境やシミュレーションなどで研究の評価をされている方であったため，チュートリアルの目指す目的と一致する方が多かった．

チュートリアル後に，それぞれのテーマの実験毎に，講義内容へのフィードバック，各実験ツールへの改善要望，一般的なネットワーク実験に関する議論などを実施し，チュートリアル運営側にも収穫のある内容となった．

また，チュートリアル後に実施したアンケートでは，講義内容の習熟度合いは参加者全員から「理解できた」という反応があり，また 1/3 の参加者からは今後 StarBED 利用し実験をしてみたい，という意見を頂いた．

5.3.1 BGP エミュレーション

BGP エミュレーションチュートリアルでは，AnyBed [4] を用いた BGP Emulation の基本をハンズオン形式で実施した．図 8 はチュートリアルの様子である．参加者は 5 名であった．1 人当たり 2 台の UCS サーバを用い，KVM を用いて 100 台の仮想マシンを使い，100AS リングトポロジーを作成して BGP

[1] http://asiafi.net/meeting/2012/summerschool/starbed-ws/

図 7: ASIA FI Summer School 午前中の発表

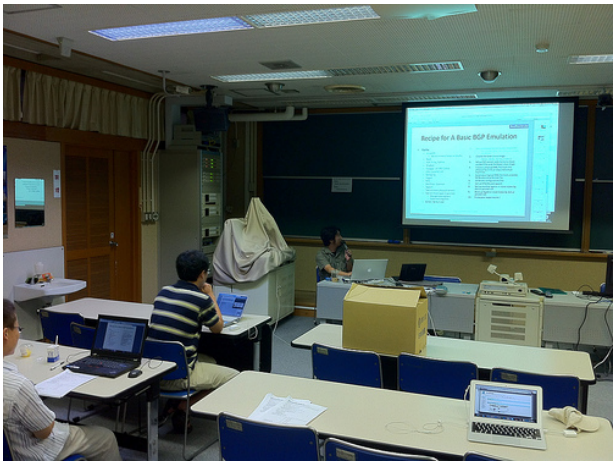

図 8: BGP Emulation チュートリアルの様子

の経路障害と伝搬の様子を検証するチュートリアルを実施した．チュートリアルの資料は，http://web.sfc.wide.ad.jp/~tazaki/asiafi12-starbed/index.php?BGP%20Emulation%20Hands-on に掲載してある．また，チュートリアルの様子は Ustream 上（http://www.ustream.tv/recorded/24900577/highlight/288157）にビデオ録画を残してある．詳細に関してはそれぞれの資料を参照いただきたい．

5.3.2 無線 エミュレーション

有線ネットワーク上に仮想的に無線ネットワークを構築し，有線ネットワーク向けのアプリケーションを検証するために QOMET [5] の開発が進められている．

本チュートリアルではこの QOMET を利用し，簡単な模倣無線環境を構築する方法についての手順を実際に QOMET の設定を行いながら確認した．本チュートリアルは京都大学にて行われたが，環境自体は StarBED 上に構築され，ここに対して各参加者が VPN 接続をし，実際にノードの設定を行った．StarBED 上には仮想ノードを利用して，一環境あたり 20 台の仮想ノードを動作させ，QOMET を用いてそれぞれのノード間接続に無線環境の特性を取り入れた．参加者は 10 名だったため，1 チーム 2 人の 5 チームを作成し，2 人で相談しながら環境の構築を進めて貰った．

チュートリアルとしては，最初に QOMET の概要と仕組みについて話した後，以下の 3 つのタスクを設定し，順番にそれを進めて行き，問題があればチューターが対応する形式で行われた．

1. **（用意された設定ファイルを用いて）静的な OSLR で制御された無線環境を構築，構築された環境を GoogleEarth を用いたビジュアライザで確認**

   本段階では参加者は QOMET の設定を特に行わずに，用意された設定を確認し，その設定から構築される環境をビジュアライザで確認することにより，直感的に設定の概要を把握することを目的にしている．また，この段階で設定ファイルを構築した後の基本的な QOMET の起動方法を習得する．

2. **タスク 1 で構築した環境のパラメータを変更しネットワーク環境の変化を確認する**

   それぞれの参加者が変更を行った設定ファイルを利用し，タスク 1 で確認したトポロジと比較することで，どのように環境に変化が生まれるかをビジュアライザを利用して確認することで，QOMET パラメータの影響の範囲を確認する．

3. **モビリティを導入**

   タスク 1 および 2 ではノードが動かない静的な環境の模倣であったが，ここで，ノードに動作モデルを適用し，モバイルノードの動作と OLSR で構築されるネットワークトポロジの変化について観測を行った．

これらのタスク群で利用したビジュアライザの例を図 9 に示す．

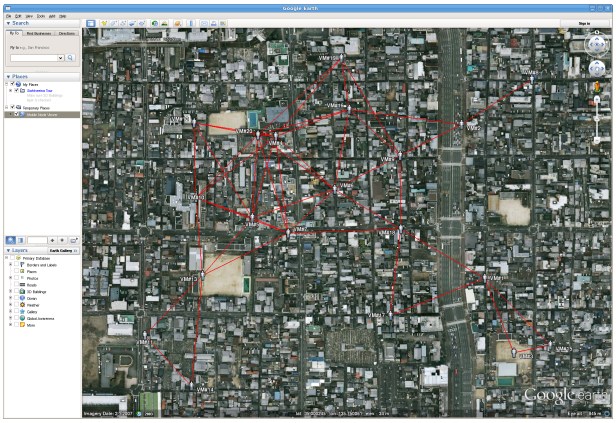

図 9: QOMET による無線環境

本チュートリアルでは参加者により非常に活発に作業が行われた．結果，多くの参加者が有用であったとのアンケート回答を寄せた．

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．

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] 知念賢一, 宮地利幸, 篠田陽一, *“Kuroyuri:* **ネットワーク実験記述言語処理系***”*, コンピュータソフトウェア, 第 27 巻, 第 4 号, pp. 43-57, 日本ソフトウェア科学会, 2010 年, 11 月.

[3] Simplified Wrapper and Interface Generator (online), `http://www.swig.org/`.

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

[5] Razvan Beuran, Junya Nakata, Takashi Okada, Lan Tien Nguyen, Yasuo Tan and Yoichi Shinoda, *“A Multi-purpose Wireless Network Emulator: QOMET”*, *AINA 2008*, pp. 223-228, Mar, 2008.