# Title: Deep Space One Working Group 2003年活動報告

Author(s):

宮地利幸 (toshi-m@jaist.ac.jp)

三角真 (m-misumi@jaist.ac.jp)

知念賢一

Date: 平成16年7月30日

# 1 設立目的

インターネットは全世界にわたる巨大なネットワークへ成長を遂げた．これに従い，インターネット上で運用することを前提とした様々な技術が開発され，それが実際にインターネットへ導入されている．

従来のインターネットは実験的な面と，実用的な面とを合わせ持っていたため，インターネット上で様々な実験が行われていた．しかし，現在のインターネットは，すでに社会基盤として利用されており，同様にインターネット上で様々なサービスが社会基盤として動作している．このような社会基盤としてのインターネット上で，何らかの実験を行った場合，既存のサービスに影響をおよぼす危険性がある．新たな技術やその実装は，その動作の正しさや影響範囲が不明であるため，このようなネットワーク上では，動作させることは事実上不可能である．したがって，インターネットに類似し，かつインターネットから分離した環境 (シミュレーション環境) において，その動作の検証を行う必要がある．

このような環境を構築するために，いくつかの手法が利用されている．実際にインターネットに導入されるソフトウェアやハードウェアの動作を検証するためには，実際に導入されるソフトウェアやハードウェアそのもの (以下実ノード) による，環境上で検証を行う必要がある．

本ワーキンググループは，実ノードを用いた大規模なシミュレーション環境の構築，および，その利用についての議論を行うことを目的に設立された．

現在は，実ノードを用いた大規模なシミュレーション環境として，北陸 IT 研究開発支援センター (通称 StarBED)[1] を中心に研究活動を進めている．

# 2 StarBED

StarBED は 512 台の PC とそれらを接続するネットワーク機器および，トラフィックの解析装置からなるネットワークシミュレーションを目的とした PC クラスタである．

StarBED の概念的なトポロジを図 1 (a) に示す．シミュレーション用の 512 台のノードが用意されており，シミュレーション用と制御用のネットワークへ接続された 2 種類のインターフェイスを持っている．これにより，シミュレーション自体と，それを制御するためのトラフィックを分離することが可能であり，制御のためのトラフィックによるシミュレーションへの影響を回避できる．

利用者は，各ノードが接続されているクロスコネクトスイッチの VLAN[5] の設定を変更することで，シミュレーションに必要なトポロジを構成することができる．図 1 (b) および図 1 (c) は，StarBED の物理ネットワーク上に，どのようにシミュレーションのためのトポロジが構築されるかを示している．たとえば，図 1 (c) 中には，3 つのネットワークが存在し，それぞれ Node1 と Node2 と Node3，Node2 と Node4，Node3 と Node4 が属する．これらのネットワークは図 1 (b) 中のクロスコネクタの VLAN または VP/VC の設定によって仮想的に構成され，その結果，図 1 (c) のような VLAN が構築される．

表 1: StarBED のクライアント装置

| クライアント装置 | 台数 | インターフェース | |
|---|---|---|---|
| | | Ethernet | ATM |
| A | 208 | 0 | 1 |
| B | 64 | 1 | 1 |
| C | 32 | 4 | 1 |
| D | 144 | 1 | 0 |
| E | 64 | 1 | 0 |

表 2: StarBED のクライアント装置

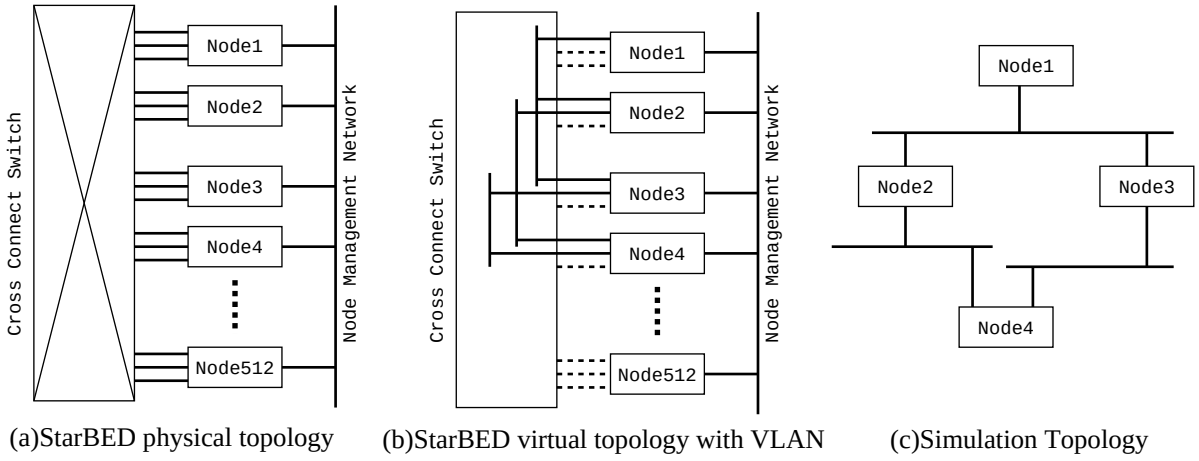
| OS | Size (Gbyte) |
|---|---|
| Windows 2000 Server | 4 |
| FreeBSD 4.4 Release | 4 |
| データ格納用空きスペース | 2 or 9 |
| Turbolinux 7 Workstation | 16.9 |

StarBED の PC(ノード) はそのインターフェースの種類などから，A から E までの 5 つのこれをクライアント装置と呼ばれるグループに分けられる．利用者はクライアント装置ごとに貸し出しを受け，時分割で利用する．各クライアント装置に所属するノードに搭載されたシミュレーション用のインターフェースの種類，ノードの台数を表 1 に示す．

各ノードのディスクは 4 つのパーティションに区分されている．ディスクのパーティション構成を表 2 に示す．

ノードは標準で PXE (Preboot eXecution Environment)[2] を用いて起動し，取得したブートローダにより，起動する OS (ハードディスク内のパーティション) を選択する．利用者は表 2 の 3 種類の OS，あるいは別の OS をインストールして利用することも可能であるが，使用後は元の環境に戻す必要がある．

シミュレーション用のノードが 512 台で足りない場合は，仮想ノードを利用し，一台の実ノードを数台に多重化することも可能である．仮想ノードを実現するソフトウェアとして，i386 アーキテクチャの PC をエミュレートする米 VMWare 社の VMWare を用いる．ただし，この方法を用いた場合，インターネットとの類似性が損なわれてしまう場合があるため，実施する実験の特性により，利用可能であるかを判断しなくてはならない．なお，




(a)StarBED physical topology (b)StarBED virtual topology with VLAN (c)Simulation Topology

図 1: StarBED の概念的トポロジとシミュレーション環境への適応

StarBED の 1 台の実ノードは，最大 10 多重できるように設計されており，最大 5120 台規模のシミュレーション環境を構築することが可能である．

StarBED には利用者用のラックが用意されており，利用者が持ち込んだネットワーク機器などを格納して，シミュレーションに参加させることが可能である．利用者用ラックのシミュレーション用のスイッチへの接続メディアは Ethernet 100BaseTX である．

# 3 現在の研究内容

シミュレーション環境を構築するための方法について議論および実装を行っている．これは，一般的にシミュレーションを行う際に，シミュレーション環境を容易に構築するツール群が必要との意見がもっとも多かったためである．

現状の研究および開発のトピックは以下の 2 点である．

- シミュレーション環境の設定記述言語
- 設定記述言語に従ったノードおよびスイッチの設定プログラム

我々の実装したシステムは，ConfigCoordinator，StarBed Resource Manager(SBRM)，Node Supervisor System(NSS) および SwitchConf で構成され，以下のように動作する (図 2).

1) ConfigCoordinator が利用者が記述した言語を読み込み，SBRM に対して，適切なノードをシミュレーション用のノード (以下シミュレーションノード) に割当て要求を発行する．

2) ノード割り当て要求を受け取った SBRM は，ノードに必要されているネットワークインターフェースの数とメディアから，適切なノードを ConfigCoordinator に通知する．この時 SBRM は ConfigCoordinator に通知したノードを，使用中ノードとして記録しておく．

3) シミュレーションに必要なすべてのノードの情報を得た ConfigCoordinator は，実際にノードおよびスイッチへ設定を行う NSS と SwitchConf の設定を生成する．

4) NSS および SwitchConf はこの設定ファイルに基づいて，ノードおよびスイッチの設定を行う．

## 3.1 ConfigCoordinator

現在，プロトタイプを実装している．また，適切な設定ファイルの形式について検討している．

SBRM との通信および，NSS，SwitchConf の設定ファイルの生成のため，ConfigCoordinator の設定ファイルには以下のような項目が必要である．

- プロジェクト名
- 責任者およびその連絡先
- ネットワーク設定
  - ネットワーク名
  - メディア
- ノード設定
  - ノード名
  - OS の種類
  - OS イメージ
  - ブートの方法 (ディスクレス or ディスクを利用する)
  - ネットワークインターフェースの数およびメディア
  - ネットワークインターフェースの所属するネットワーク
  - それぞれのネットワークインターフェースの IP アドレス

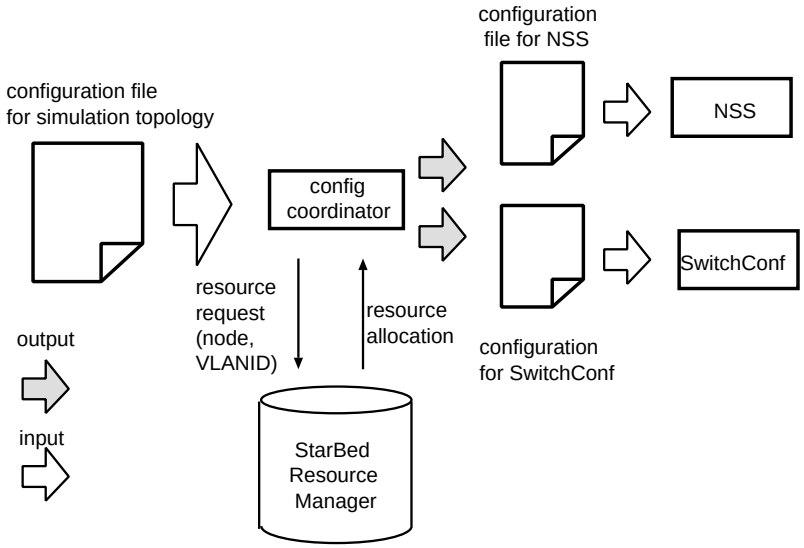



図 2: シミュレーション環境構築までの流れ

また，ノード数が大きくなれば，その記述もコストが高くなるため，オブジェクト指向でいうところのクラスと，インスタンスのような関係も採入れることを視野に入れている．

## 3.2 NSS

NSS は，シミュレーション用のノードへ適切な設定を行うためのシステムである．設定は，OS の導入と IP アドレスやホスト名などの各ノードの固有の設定に分かれる．

### 3.2.1 OS の導入

前述の通り，StarBED では，シミュレーション終了後に利用したノードを初期状態に戻しておく必要がある．ハードディスクへ変更を加えること無くシミュレーションを行えば，このコストを回避することができる．そこで，ディスクレスで動作する OS でシュミレーションを行う形態を設けた．この場合，Trivial File Transfer Protocol(TFTP)[4] を利用して OS の動作に必要なカーネルとディスクイメージを取得する．ただし，TFTP の仕様上，ファイルサイズは 32MBytes 未満に制限されているため，ディスクレスで動作する OS には大きな制約となる．

なお，この TFTP のファイルサイズの上限は，ブロック番号が 2 バイト (16 ビット) で表現され，ブロックサイズが 512 バイトであることに起因する．すなわち，$2^{16} \times 512 = 2^{16+9} = 32\text{M}$ である．

ハードディスクを利用する OS を新たに導入する場合は，前もって，あるノードで雛形となるディスクイメージを作成し，そのパーティションの内容をファイルサーバに保存する．そして，対象ノードでファイルサーバから取り出したディスクイメージをハードディスクに書き込み，複製する．ただし，OS の動作に利用されているディスクに書き込むことは困難であるため，ディスクへの書き込みはディスクレスで動作する OS をネットワークブートした後に行う．

パーティション単位のディスクイメージ配布の概略を図 3 に示す．ディスクへの読み書きは dd，ファイルの転送には File Transfer Protocol(FTP)[3] を用いる．また，データを圧縮する手法として gzip を利用している．

### 3.2.2 ノード個別の設定

OS 導入後，各ノードではその役割に応じたアプリケーションの起動や IP アドレスなどの割り当て等，ノード個別の設定を行う場合が多い．そこで，以下のようなファイル群をテープアーカイブ (tar) 形式で集め，各ノードへ転送することにより，ノード個別の設定を行う．この tar 形式のファイルは標準で導入された OS に対しての，差分を設定するためのものであるため，差分ファイルと呼ぶことにする．差分ファイルには以下のものが含まれる．

- アプリケーションの実行ファイル
- アプリケーションの起動スクリプト
- OS に用意されている設定ファイル
- ノード個別の設定を行うためのスクリプト

あるノードで，あるアプリケーションが起動している必要がある場合は，差分ファイルにアプリケーションの実行ファイルを含めておき，その実行ファイルを必要な位置に複製した後，実行ファイルをあるタイミングで起動するための設定を OS に対して行う．

IP アドレスなどの設定については，多くの場合 OS に設定ファイルが用意されていることが多いため，このファイルの置換，または必要な設定を追加後，必要なコマンドの実行や，場合によってはノードの再起動により設定を行う．

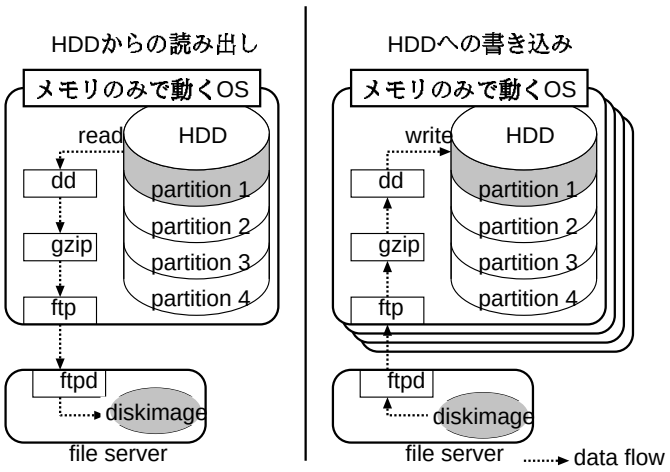



図 3: ハードディスクへの OS 導入

また，差分ファイルに含まれるノード個別の設定を行うためのスクリプトを実行することでも，各種の設定や行うことができる．例えば IP アドレスなどは，ifconfig など専用のコマンドにより設定が可能なことが多いため，このようなコマンド群を呼び出すスクリプトを用意，実行することで設定可能である．

#### 3.2.3 システム構成

NSS は，管理ホストで動作する Node Manager (NM) と，シミュレーション用ノードで動作する Node Agent (NA) からなる．NM は与えられた設定に従い，ノードで動作している NA に対し，ノードへの OS の配布及びノード個別のパラメータ設定に関する指示を出す．それを受け，NA がそれらの処理を実行する．

NSS がハードディスクを用いるノードに対し，OS の導入及び個別のパラメータの設定を行う一連の流れを図 4 に示す.

1) NM は，ディスクイメージを書き込むノードを Wake on LAN を用いて起動させる．

2) ハードディスクを用いるノードはディスクイメージを導入するため，まずディスクレスで動作する OS で起動する．このため，ディスクレスで動作する OS のためのブートローダを PXE が TFTP を利用して取得する．

3) 次に，ノードのブートローダがディスクレスで動作する OS に必要なカーネルとディスクイメージを取得する．

4) ディスクイメージを書き込むためのディスクレスで動作する OS が起動すると，そのノード上で動作している NA から，NM に対して起動したことが報告される．

5) この報告を受け取った NM は，書き込むパーティション位置とファイルを NA に対して指示する．

6) NA は，それに従いハードディスクにデータを書き込む．

7) ディスクイメージの書き込み終了後，ノードは再起動し，先ほどハードディスクに書き込んだパーティションからノードを起動するためのブートローダを取得する．

8) ハードディスクを用いる OS の起動後，その上で動作している NA が OS が起動したことを NM に報告する．

9) NM は，その報告を受け取ると，NA に対し差分ファイルを取得して適用するよう指示する．

### 3.3 SwitchConf

SwitchConf は，ConfigCoorinator から与えられた設定に従い，StarBED のシミュレーション用スイッチに TELNET プロトコルで接続し，必要な設定を行う．現在は StarBED 内の CISCO Catalyst に特化した簡単なスクリプトとなっているが，既に様々なコミュニティから配布されているスイッチ設定用のプログラムなどの利用も可能である．

### 3.4 SBRM: StarBED Resource Manager

SBRM は StarBED 内のリソースを管理するプログラムである．リソース情報の提供と，利用の排他処理を行う．現在の管理対象はノードと VLAN-ID である．SBRM はリソースの詳細情報と利用状態を記憶し，ConfigCoordinator の割り当て要求に応じて，利用状況を変更する．この際，既に利用されている場合には，利用不能であることを報告する．

一般に，リソース利用者側ではリソース情報取得と資源獲得を行うが，競合者が多い場合には，取得したリソー

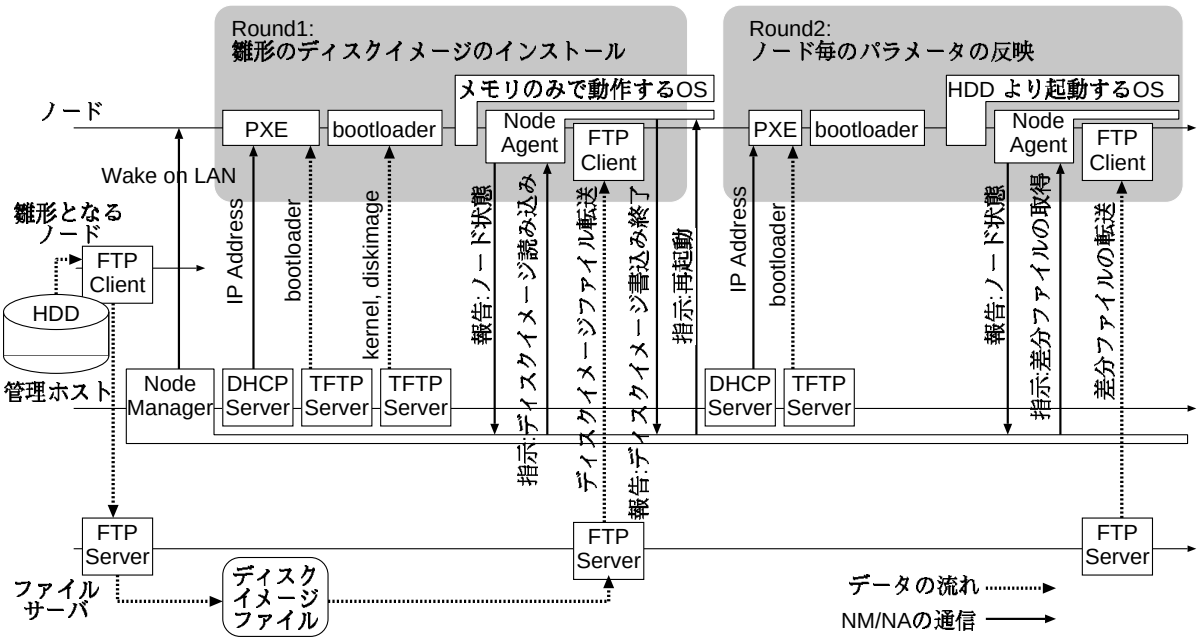



図 4: ハードディスクを用いる OS の動作するノードへの OS 配布

ス情報が，資源獲得の時点で既に変更されおり，必要な資源を獲得できない状況がおきうる．このような事態を避けるため，SBRM ではその二つをアトミックに処理する形態を導入した．具体的にはリソース検索と獲得を統合した．この検索では，インターフェースのメディアと数が記述できる．

なお，ConfigCoordinator や NSS は StarBED で行われているシミュレーションの数の分だけ起動されるが，SBRM は全体を管理するプログラムであるため，StarBED で一つだけ起動される．

### 3.4.1 SBRP: StarBED Resource Protocol

SBRM との通信には SBRP(StarBED Resource Protocol) と呼ばれるプロトコルを用いる．SBRP は一般的なテキスト形式のプロトコルである．表 3 にコマンド一覧を示す．

たとえば，一つのノードを使う際の ConfigCoodinator からの通信内容は以下のようになる．

| | |
|---|---|
| `LIST node free` | • 一覧を取得 |
| `INFO node n13` | • ノード n13 の情報を取得<br>• 一覧内容から n13 を使うことを決める |
| `HOLD node n13` | • n13 を確保<br>• n13 を何かの役割に使う |
| `RELEASE node n13` | • n13 を解放 |

ここで，ノード n13 が使えず，代わりに n54 が使える場合は，以下のようになるだろう．

| | |
|---|---|
| `INFO node n54` | • ノード n54 の情報を取得<br>• 一覧内容から n54 を使うことを決める |
| `HOLD node n54` | • n54 を確保<br>• n54 を何かの役割に使う |
| `RELEASE node n54` | • n54 を解放 |

前述の検索と獲得を統合した形態では，FINDHOLD でノードの選択を SBRM にまかせて，得られたノードを使うことになる．以下の例は，ノード n54 から n57 を得て利用するケースである．

| | |
|---|---|
| `FINDHOLD node num=4,⇒`<br>`if[media=fastethernet],if` | • FastEthernet ともう一つの NIC を搭載したノードを 4 つ要求<br>• 得られた n54..n57 を何かの役割に使う |
| `RELEASE node n54` | • n54 を解放 |
| `RELEASE node n55` | • n56 を解放 |
| `RELEASE node n56` | • n57 を解放 |
| `RELEASE node n57` | • n58 を解放 |

FINDHOLD を使えば，利用できないノードを確保する可能性がなくなり，再確保のオーバヘッドを省くことができる．

## 4 StarBED の利用

今年度は，IP traceback や Multicast, および Atomized BGP 等の実験が StarBED で実施された．

表 3: SBRP コマンド一覧

| | |
|---|---|
| リソース情報取得 | |
| LIST | 一覧 |
| INFO | 詳細情報 |
| リソース獲得/解放 | |
| HOLD | 獲得 |
| FIND | 検索 |
| FINDHOLD | 検索と獲得 |
| RELEASE | 解放 |
| バージョン問い合わせ | |
| VER | プロトコルバージョン |
| SYST | 相手プログラムバージョン |
| セッション処理 | |
| JOIN | セッション参加 |
| 利用者簡便 | |
| HELP | ヘルプ情報 |
| NOP | 無処理 |

# 5 今後の活動

短期的な今後の研究内容は以下の通りである.

- ConfigCoordinator のプロトタイプを完成させ，利用者が利用しやすい記述言語についての議論
- シミュレーションを利用者のシナリオに基づいて進行させる方法
- シナリオ記述言語と ConfigCoordinator 記述言語の親和性を検討

また，StarBED の利用者をサポートすることにより，実際に利用した経験からの，必要な機能についてのヒアリングなどを行う.

# 参考文献


[1] 通信放送機構 北陸 IT 開発支援センター. http://www.hokuriku-it.tao.go.jp/.

[2] Intel Corporation. *Preboot Execution Environment (PXE) Specifiction Version 2.1*, sep 1990.

[3] J. Postel and J.K. Reynolds. File Transfer Protocol, RFC959. October 1985.

[4] K. Sollins. The TFTP Protocol (Revision 2), RFC1350. July 1992.

[5] IEEE standard for Local and Metropolitan Area Networks: Virtual Bridged Local Area Networks. December 1998.