2004－DPS－116　(17)
2004／1／30

実ノードを利用したネットワークシミュレーションにおけるノードへのOSの導入及びパラメータ設定機構の開発

三角　真[†]　宮地利幸[†]
知念賢一[†]　篠田陽一[††]

インターネットで実際に利用されている機器を用いて，インターネットをシミュレーションすることができる施設として北陸IT研究開発支援センター (StarBED) がある．このような施設で，実ノードを用いた大規模なネットワークシミュレーションを行う際には，そこに存在するノードの数も増え，ノードやネットワークの設定に多くの人的資源が費やされる．このようなシミュレーションを行うための環境の構築を，より容易に行うことができれば，ネットワークシミュレーション自体に，より多くの人的資源を割当てることができる．

StarBEDには既に各ノードに対して，OSを導入する仕組が存在するが，利用者は常にそのシステムを操作する必要があり，人的資源を節約するといった面では，有用であるとは言えない．そこで，我々は，ノードへのOSの導入やネットワーク設定および，ノードに個別なパラメータの設定を自動的に行う機構である，Node Supervisor System (NSS) の設計と開発を行った．NSSにより，人的な資源が必要となる作業をまとめ，利用者の拘束時間を短縮することにより，人的資源の削減を図る．

本論文では，NSSの設計とその実装について述べ，既存のStarBEDへのOSの配布システムと比較しNSSの評価を行う．

NSS: Node Supervisor System on StarBED

— distributing OS to nodes and setting up parameters on each node

Makoto Misumi,[†] Toshiyuki Miyachi,[†] Ken'ichi Chinen[†]
and Yoichi Shinoda [††]

Hokuriku IT Open Laboratory (StarBED) is the equipment which can perform the simulation of the Internet using the apparatus actually used by the Internet. With such equipment, in case the large-scale network simulation using the real node is performed, the number of the nodes which exist there also increases and many human resources are spent on a setup of nodes and network. If environment for performing such a simulation can be built more easily, more human resources can be assigned to the network simulation itself. StarBED already has the system which distribute OS to each node. However, since a user always needs to operate the system, it is not useful in the field of saving human resources. To avoid these problems, we designed and developed Node Supervisor System (NSS). NSS performs installation of OS, and characteristic setup of each node. NSS aims at curtailment of human resources by gathering the work for which human resources are needed, and shortening a user's actual working hours.

This paper describes a design and its implementation of NSS, and NSS is evaluated as compared with the distribution system of OS to the existing StarBED.

1. はじめに

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，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そのもの (実ノード) を利用してインターネットを模倣した環境 (ネットワークシミュレーション環境) を構築し，さらにその上で実際にインターネットへ導入される実装を被検証対象として評価を行わなければならない．

実ノードを用いネットワークシミュレーション環境

† 北陸先端科学技術大学院大学 情報科学研究科
School of Information Science, Japan Advanced Institute of Science and Technology
†† 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

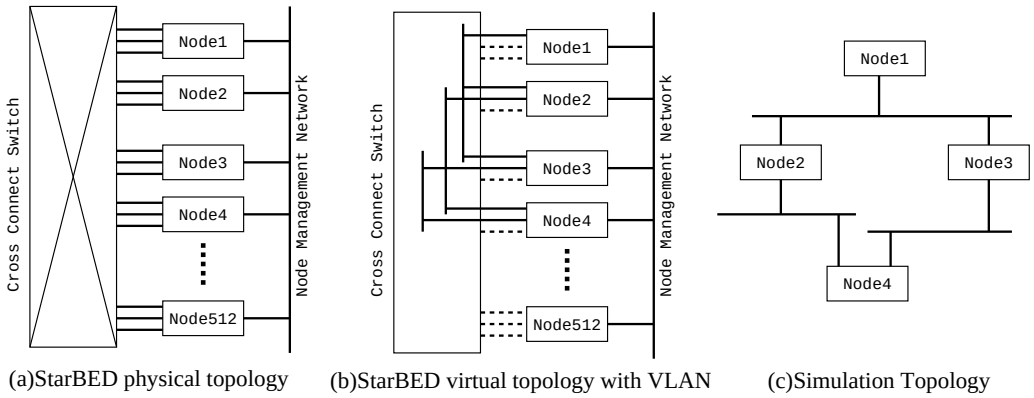

図 1 StarBED の概念的トポロジとシミュレーション環境への適応

を構築する際に，多くのノードを適切に設定する必要がある．ネットワークシミュレーションの規模が大きくなるに従い，設定が必要なノードの数も増加し，シミュレーション環境の構築に多くの人的資源を費やさなければならない．そこで本研究では，実ノードを用いた大規模ネットワークシミュレーション環境構築の自動化を行うことにより，それに要する人的資源の削減を図る．

2. 実ノードを利用した大規模ネットワークシミュレーション施設

大規模なネットワークシミュレーション環境を構築する際には，多くのシミュレーション環境の要素である PC など (ノード) と，ネットワーク機器に対し操作を行う必要がある．シミュレーションに要するノードの数が増えるにつれ，IP アドレスのような個別のノードに対する特別な設定も増え，ネットワークシミュレーションに用いるネットワーク機器への設定も複雑になる．

2.1 StarBED

実ノードを用いて大規模なネットワークシミュレーションを実現する施設として，StarBED がある [1]．図 1 に StarBED の概念的なトポロジと，実際に物理トポロジ上にどのようにシミュレーション環境が構築されるかを示した．図 1 (a) が StarBED の概念的なトポロジである．シミュレーション用の 512 台のノードが用意されており，シミュレーション専用のネットワークと制御用のネットワークへ接続されたインタフェイスを持っている．

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環境が構築されるかを示している．利用者は各ノードのシミュレーション用のネットワークインタフェイスが収容されているクロスコネクトスイッチの VLAN の設定を変更することで，必要なトポロジを構成する．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 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．

StarBED のノードには，標準でいくつかの OS がインストールされており，利用者はこの OS を利用することも可能である．ノードは標準で PXE [2] により起動し，Hard Disk Drive (HDD) 内のいずれの OS を用いるかは，PXE にて取得するブートローダで選択している．利用者は，施設をグループ単位で時分割利用しており，ネットワークシミュレーション終了後，標準状態に戻す必要がある．

利用者がシミュレーション用にカスタマイズした OS を利用する場合には，それらを必要な台数のノードへ導入する必要がある．また，それらを標準状態へ復帰させる際にも，利用したノードすべてに対して操作を行わなければならない．StarBED を利用し数百台規模のシミュレーション環境を構築した場合には，OS の導入や，標準状態への復帰という作業は非常に大きなコストを必要とする．

2.2 標準状態復帰システムの問題点

現在，StarBED にはノードの HDD を標準状態に戻す手段が用意されている．グループ毎に標準状態のパーティション単位で用意されたディスクイメージをネットワークを介してコピーするもので，その手順は，以下のようになる．

(1) シミュレーションで利用する OS を 1 台のノードに導入し，雛形とするノードを作成．
(2) 雛形とするノードのディスクイメージを作成するためのアプリケーションが動作する Floppy Disk (FD) を挿入．
(3) 取得したパーティションのディスクイメージを，ネットワーク経由でファイルサーバに保存．
(4) ディスクイメージ配布プログラムを動作させるための配布先ノードへ FD 挿入
(5) 個々のノードを操作し，任意のディスクイメージをどのパーティションに導入するかを選択．
(6) FD を取出し，ノードを再起動．
(7) ディスクイメージの配布修了後，ノードの個別設定．

また，このシステムは，人間が逐次 GUI により操作することしかできないため，ディスクイメージ及びパーティション選択の自動化は不可能である．この既存の標準状態復帰システムは単一のノードに対しては有用であるが，数百台規模のノードに対してこのシステムを利用するのは，同等の操作を複数のノードに対して行う必要があるためコストが大きい．また，起動に FD を利用することから物理的にノードにアクセスする必要があり，多くの時間と，労力を要する．

本論文では，ネットワークシミュレーション環境を構築する際に，人が拘束される時間を人的資源の尺度とする．

既存のシステムを利用すると，利用者はノードに対して，雛形ノードのディスクイメージを導入する際には，OS のコピー設定の時点から，個別の設定を行うまで，連続的に操作を要求される．したがって，人的資源の観点からは非常にコストが大きいと言える．

我々は，ノード設定の自動化を図り，人的資源の削減を図るために Node Supervisor System を提案し実装した．

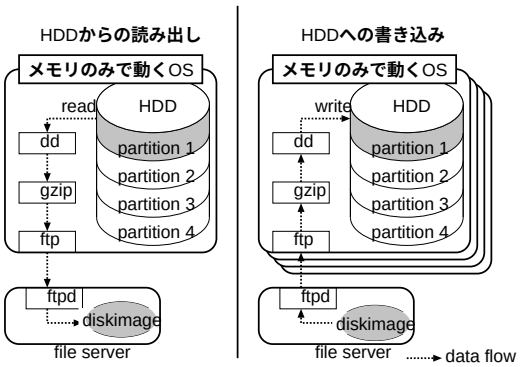

図 2 HDD への OS 導入

3. NSS の設計

本節では，Node Supervisor System (NSS) の設計について述べる．

3.1 ノードへの OS の配布

ノードへ OS を導入する際に，複数のノード間で利用する OS が同一であれば，OS の基本部分を構成するファイル群と，個々のノード間で異なるパラメータを設定するためのファイル群に分けて管理する．ここでパラメータとは，該当ノードで，動作していなければならないアプリケーションや，IP アドレスやホスト名などのノード固有の設定を指す．これらを，別に管理することで，保存しておかなければならないファイルの合計サイズなどが小さくなり，ファイルサーバに必要なディスク容量や，転送しなければならないファイルサイズが小さくなるというメリットがある．したがって，各ノードに基本となる OS を導入後，各ノードで個別の設定を行うファイル群をその上に適用し，個別のパラメータを設定を行う方法を採用した．

各ノードに個別のパラメータを設定を行うためのファイル群には，各 OS に用意されている設定ファイルや，コマンドを実行し各パラメータを設定するためのシェルスクリプトおよび，様々なアプリケーションの本体やその設定ファイルが含まれる．これらのファイルの実行または，特定の位置に配置することで，ノード個別のパラメータの設定を行う．

3.2 メモリのみ (HDD 無し) で動く OS の導入

前述の通り StarBED では，利用したノードはシミュレーション終了後に初期状態に戻しておく必要がある．したがって，HDD を用いると，初期状態に戻すとい

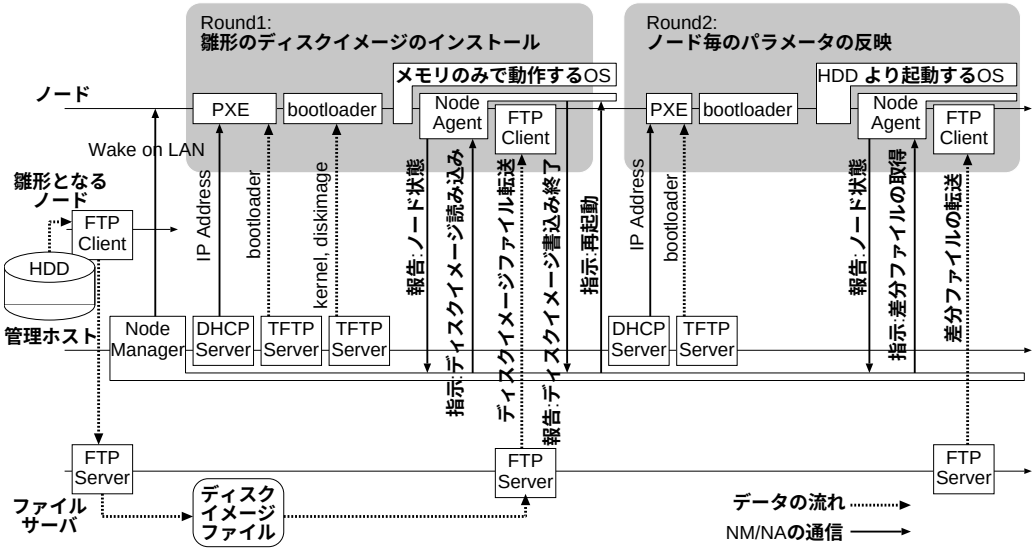

図 3 NSS による HDD を用いて動作するノードへの OS 配布

うコストがかかる．また，標準で用意されている OS 以外の OS を利用する場合も，OS の導入というオーバーヘッドは非常に大きい．

HDD の利用なしに，シミュレーションを行うため，メモリのみで動く OS を用意した．この手法を用いた場合にはノードは起動する際に 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．TFTP [4] は，ファイルを転送する際そのファイルをブロック毎に転送する．TFTP のブロック番号は，2bytes で表されるので 65535 のブロック数をカウントすることが出来る．512bytes が TFTP のブロックの単位であるので，TFTP で転送できる最大のファイルサイズは 65535blocks × 512bytes = 33553920bytes ≒ 32Mbytes 程度となる．よって，この上限以下のディスク容量で十分なノードならばこの手法は有用であるが，より大きなディスクスペースを要求するノードの場合は HDD を用いなければならない．

メモリのみで動くシステムを利用する際には，起動時に用いるカーネルとシミュレーションに必要な機能を持つディスクイメージを，あらかじめ用意しておく必要がある．個々のノードに依存するファイルは，OS の起動後 FTP [3] を用いて取得する．

3.3 HDD を用いて動作する OS の導入

HDD を用いるノードに対し OS を導入する際には，雛形のノードを用意し，そのディスクイメージを取得し，それを対象ノードへ配布する．

HDD に読み書きを行うにあたり，HDD のディスクイメージを取得し HDD のパーティションへの書き込みを実現するシステム動作は HDD から独立したもので無くてはならない．そのため，メモリのみで動く OS を，ネットワークブートすることで HDD から独立した環境を実現する．この，メモリのみで動作する OS を用いてディスクの読み書きを行う．

3.4 ノードへの OS 導入の手順

ノードへ OS を導入する際に必要な設定作業を前もってしておくことで，OS の導入が終了するまで，人間がその場に拘束されないような手順を採用した．この手順を以下に示す．

(1) シミュレーションで利用する OS を 1 台のノードに導入し，雛形となるノードを作成．
(2) 上述のノードで，ディスクイメージを作成するための OS を動作させるため，メモリのみで動く OS が起動するように設定．
(3) ディスクイメージを作成しファイルサーバ転送．
(4) NSS の設定ファイルを生成．
(5) NSS の実行．

4. NSS の実装

ここでは，前述の設計にしたがって，実際に行った実装についての詳細を述べる．

4.1 HDD を用いて動作する OS のための実装

メモリのみで動く OS 上で，dd コマンド，ftp コマンド及び gzip コマンドを利用し，パーティション単位のディスクイメージ配布を実現する．

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 2 に示す．指定された HDD のパーティションのディスクイメージを読み取る場合，dd で読み取ったデータを gzip で圧縮しつつ ftp を用いてそのディスクイメージをファイルサーバに保存する．一方，HDD にディスクイメージを書き込む場合，FTP サーバから取得した gzip により圧縮されたディスクイメージを伸長しながら，dd で HDD の任意のパーティションにデータを書き込む．

4.2 NSS の動作説明

NSS は，管理ホストで動作する Node Manager (NM) と，ノードで動作する Node Agent (NA) からなる．

NM はノード毎の起動する方式，OS の種類，OS を導入するパーティション，HDD に導入するディスクイメージファイル名，メモリのみで動作させる際に用いるカーネルのファイル名等，前もって利用者により設定された内容から，ノードで動く NA に対し指示を出し，ノードへの OS の配布，ノード個別のパラメータを設定する．

また，NM はシミュレーションに用いるノードの状態を保持し，ノードで動作する NA に対し指示を出す．これにより，ノードへの OS 導入の自動化を実現する．

NSS が HDD 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 3 に示す．NM は，HDD の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．HDD を用いるノードはディスクイメージを導入するため，まずメモリのみで動作する OS で起動し，メモリのみで動作する OS を起動するためのブートローダを PXE が TFTP を利用して取得する．続いてブートローダがメモリのみで動作する 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．HDD に対しディスクイメージを書き込むための OS が起動すると，その上で動作している NA から NM に対して，このことが報告される．この報告を受け取った NM は，NA に対し，どのパーティションに，どのファイルを取得し書き込むかを指示する．NA は，それに従い HDD にデータを書き込む．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 HDD 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．HDD を用いて動作する 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．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適用するよう指示を行う．

表 1 既存 OS 配布システムと NSS の比較

	StarBED 既存	NSS
起動媒体	FD	netboot
local での操作	○ (GUI)	○ (CUI)
ネットワークごしの操作	×	○
ノード個別のパラメータ設定	×	○

5. NSS の評価

ここでは，我々が提案，実装した NSS を利用した場合と，StarBED の既存の OS 配布システムの比較および NSS の評価を行う．

5.1 NSS と StarBED 既存 OS 配布システムの比較

StarBED の既存の OS 配布システムと，NSS の機能の比較を行う．この結果を表 1 に示す．

StarBED の既存の OS 配布システムは，FD を利用して起動する．そのため，利用者は OS を導入するノードに物理的にアクセスする必要がある．また，操作を行う際，個々のノード毎に操作を行う必要があり，ネットワーク経由での操作も不可能である．NSS は，ノードへの OS の導入を自動化するために，その起動はネットワーク経由で行い，また，ネットワークごしの操作を行うことで，ノードの集中管理を行う．

NSS を用いた場合，シミュレーションに用いるノードは，PXE により起動するため，個々ノードが起動の際にネットワーク経由で取得するブートローダを，ファイルを配布するサーバ側で選択可能となる．これ

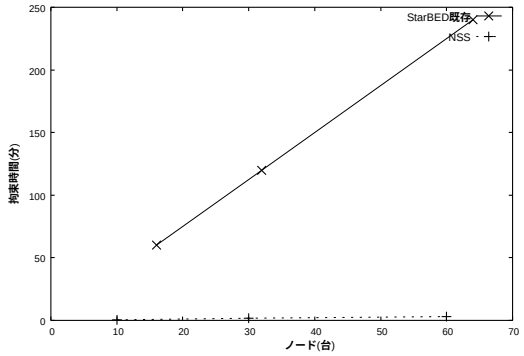

図 4 処理するノード数に対する拘束時間

により，個別ノードの HDD のいずれのパーティションから起動するか，メモリのみを用いて起動するかの選択を一元管理できる．

5.2 NSS と StarBED 既存 OS 配布システムが要する時間の比較

NSS を利用した場合と，StarBED の既存 OS 配布システムを利用した場合の，人間の拘束時間を計測した．両システムでも，それぞれのシステムが用いるディスクイメージは既に存在するものとする．それぞれのディスクイメージをノードの HDD に書き込み，ノードの設定を行うまでの過程について計測を行った．なお，個々のノードに対し書き込むディスクイメージはそれぞれのシステム毎に全ノードで同様のディスクイメージを用いた．NSS のディスクイメージファイルは 4GB のものを 1.5GB に圧縮されたものを用いた．同様に，StarBED 既存の OS 配布システムのディスクイメージファイルは，4GB のパーティションのものが 1.5GB に圧縮されているものを用いた．NSS の場合は NSS の設定ファイルを利用者が記述し，NM を実行するまでの時間を拘束時間とし，StarBED 既存の OS 配布システムは FD 挿入から全てのノード個別の設定が終わるまでを拘束時間とした．その際の拘束時間の結果を図 4 に示す．

6. 考　　察

HDD を用いるノードの構築について StarBED 既存の OS 配布システムと NSS で比較を行った．計測結果をみると，設定を行う利用者の拘束時間は既存の StarBED の OS 配布システムと NSS を用いた場合の両システムとも，扱うノードの数が増えれば増加する．しかし，図 4 からわかるように，その増加の傾きが既存の StarBED の OS 配布システムと比べ NSS の方が小さい．これは，NSS が人の判断を要する部分を初期の NSS の設定を作成する個所に集中させ，一定の処理毎に (HDD へのディスクイメージ導入修了後のノードの再起動等) 人の判断を仰がずともよいためである．これにより，StarBED 既存の OS 配布システムに比べ，NSS を利用した場合はユーザが拘束される時間を大幅に節減できることが，明らかである．

7. 今後の課題

現在の NSS は，ノード共通部分の転送に FTP を用いているため，その転送はユニキャストである．そのため，ネットワーク帯域の上限の都合上，現状 3 並列が上限となる．

シミュレーション環境の設定に要する時間の短縮を実現する一つの手法として共通部分配布のマルチキャスト化を行う．

8. おわりに

本論文では，実ノードを用いたシミュレーション環境の構築は，その規模とともに困難になることをまず確認した．その上で，実ノードを用いて数百台規模のシミュレーション環境を構築するための施設である，StarBED について紹介し，そこで利用されているノードへの OS 配布システムの詳細と問題点を挙げ，これを解決するため NSS の提案および実装についての詳細を述べた．また NSS を実装し，既存のシステムとの比較を行い，その有効性を確認した．最後に，現状の NSS の問題点について考察し，これを回避する方法について今後の課題として述べた．今後は，現状の NSS の問題点を回避するために，開発を進めて行く予定である．

参 考 文 献

1) 通信・放送機構 北陸 IT 研究開発支援センター. 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.