StarBED および SpringOS の性能評価

宮地 利幸 †¶　**知念 賢一** ‡¶　**篠田 陽一** §¶

† 情報通信研究機構 北陸リサーチセンター
北陸先端科学技術大学院大学
‡ 情報科学研究科 / § 情報科学センター / ¶ インターネット研究センター

Performance Evaluation of StarBED and SpringOS

Toshiyuki Miyachi†¶　*Ken-ichi Chinen*‡¶　*Yoichi Shinoda*§¶

†HOKURIKU RESEARCH CENTER
NATIONAL INSTITUTE OF INFORMATION AND COMMUNICATIONS TECHNOLOGY
‡SCHOOL OF INFORMATION SCIENCE / §CENTER FOR INFORMATION SCIENCE /
¶INTERNET RESEARCH CENTER
JAPAN ADVANCED INSTITUTE OF SCIENCE AND TECHNOLOGY

概 要

我々はこれまで実ノードを用いて大規模な実験を行うための施設である StarBED の運用開発と、StarBED 上での実験を支援するためのソフトウェアである SpringOS の開発を行ってきた。本論文では、StarBED 上で SpringOS を利用して実験を行った場合の、ディスクコピーや実験トポロジ構築、シナリオ実行などの各段階に必要となる時間を計測した。本実験により、SpringOS は実験の実行者が用意した設定記述にしたがい、200 台以上の PC ノードの制御を行えることと、単純なシナリオであれば、200 台規模の実験を 7 時間程度で実行できることを確認した。また、ディスクレスでノードを動作させればさらに短時間での実験実行が可能である。さらに、設定記述を用意すれば、実験自体は自動的に実行されるため、実験の実行者の拘束時間を大きく削減できる。

1　はじめに

ある技術の実装を実環境に導入する前には、実環境上の既存のサービスなどに想定外の影響を与えないよう、バグを含めた挙動の検証が必要である。これには、実環境に導入される実装を、実環境にできるだけ近い環境で実際に動作させる必要がある。このような環境を構築するためには、実環境で利用されるノードにより構成された実験用の環境が必要となる。我々はこのような環境を構築するために、実験専用の多数の PC ノードが用意された実験設備である StarBED と、StarBED に存在する実験資源を操作し実験を容易に実行するための支援ソフトウェアである SpringOS を開発してきた [1]。これにより大規模な実験用の環境を構築し、実験手順（シナリオ）を自動的に実行することが可能となった。本稿では、StarBED および SpringOS を利用して実験を行う際に必要となる時間を基礎データとして取得することを目的として実施した実験と、その結果についてまとめる。

2　StarBED と SpringOS の概要と動作

まず、評価対象である StarBED および SpringOS の概要と動作をまとめる。

2.1　StarBED と SpringOS

StarBED は、多数の実験専用の実ノードとそれらを接続するスイッチノードからなる実験設備であり、情報通信研究機構 北陸リサーチセンターに設置されている。StarBED のノードは最低 2 つのネットワークインターフェースを持ち、そのうちの一つは固定

的に常時アクセスが可能な管理ネットワークに接続されている。それ以外のネットワークインターフェースは実験用に用いられ、実験の実行者が指定したトポロジを構築するための設定がなされる。実験用のネットワークには VLAN に対応したスイッチが配置されており、実験トポロジは物理的な配線を変更することなく VLAN 設定により仮想的に構築される。

SpringOS は、複数のプログラムモジュールからなるソフトウェアスイートであり、StarBED 上での実験実行を支援する。主な機能には、実験に必要な PC ノードや VLAN 番号などの実験資源の割り当て、PC ノードへのソフトウェア導入、実験トポロジの設定、シナリオの実行などがある。以上の動作は、実験の実行者による設定記述にしたがって実行される。また、これ以外にもノードの電源管理や、起動方法の変更なども行える。

2.2 SpringOS の動作

SpringOS は以下の順序で実験を実行する。

1. 実験の設定記述の読み込み
2. 実験資源の割り当て
3. ノードへのソフトウェアの導入
4. 実験用スイッチの設定
5. 実験用ノードの初期設定およびシナリオの実行

SpringOS では実験の実行者が前もって用意した実験の設定記述を読み込み、それにしたがい必要な実験資源を確保する。次に割り当てられたノードに必要なソフトウェアの導入と、設定を行うことで、実験用の環境を構築する。

SpringOS では、PC ノードへの OS の導入には、実験用ノードと同構成のノードに必要な OS や必要なアプリケーションソフトウェアを導入し、設定したハードディスクの内容をバイナリイメージとして保存、実験用ノードに導入することによるノードの複製により実現している。このため、以上の手順以前に、ディスクイメージの作成が必要である。ただし、実験や利用する OS 環境によっては、ディスクレスシステムとして実験用ノードを利用でき、この場合は、ディスクレスシステム用のイメージが必要となる。

ディスクイメージは、SpringOS で提供しているプログラム `pickup` を用いて作成できる。実験の実行者が、まえもって設定を行ったハードディスクのパーティションをイメージとして `pickup` で取得し、ファイルサーバに保存する。

また、シナリオは管理用ノードで動作する管理用プログラム（以下 `master`）と、実験用ノードで動作するクライアントプログラム（以下 `slave`）が協調して実行する。`master` が実験シナリオ全体を保持しており、各実験用ノードで動作する `slave` が実行すべきシナリオを配布する。シナリオを受け取った `slave` はその時点からシナリオ実行を開始し、他のノードとの協調が必要になった時のみ、`master` を通したメッセージパッシングにより協調を行う。

実験終了後には、資源の返却およびノードの初期化とスイッチの初期化が必要である。資源の返却には、リソースマネージャに対して資源の返却表明を行い、ノードの初期化およびスイッチの状態を初期化する。ノードは初期状態のディスクイメージを導入する方法と、初期状態に対して加えた変更を削除する方法がある。スイッチの初期化は実験に利用した VLAN 情報の削除により実現する。

SpringOS の動作の詳細については文献 [1, 2, 3] にまとめられている。

3 実験内容

観測対象の実験として、*netperf* [4] による 2 ノード間の帯域測定を採用した。この実験に必要とした時間を測定することにより、StarBED および SpringOS の性能を評価した。また、ディスクイメージの作成時間も測定した。

3.1 ディスクイメージの作成

ディスクイメージの保存は管理ネットワークを利用して実行される。今回利用した実験用ノードが接続されている管理ネットワークの物理的構成を図 1 に示す。StarBED のノードは 1 ラックに 24 台設置され、各ラックにおかれた LAN スイッチ装置 D（以下スイッチ D）に収容されている。スイッチ D は上流の LAN スイッチ装置 F（以下スイッチ F）に接続され、ノードの管理用サーバはスイッチ F に接続されている。図中のリンクはすべて FastEthernet であり、

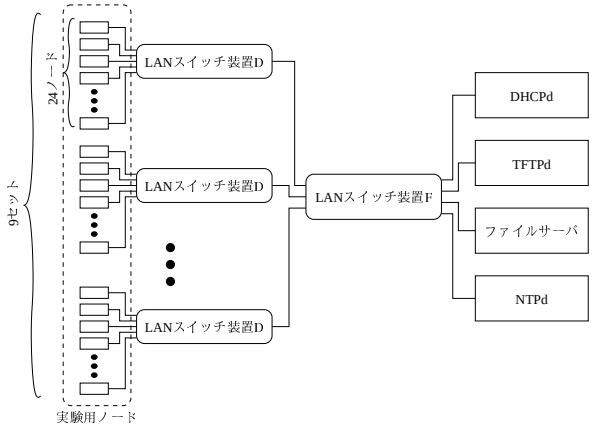

図 1: 実験に利用したノード群の管理ネットワーク構成

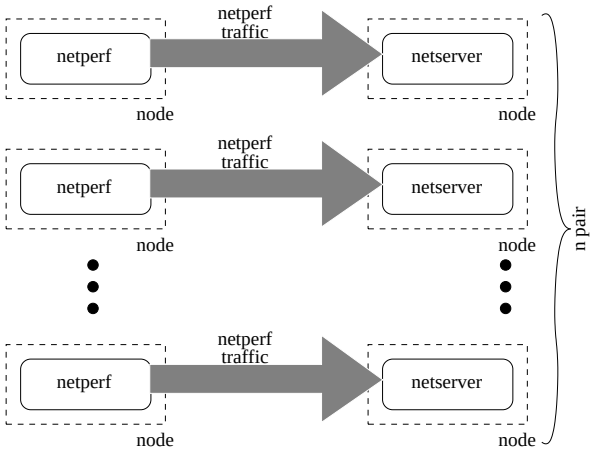

図 2: 評価実験用トポロジ

帯域は 100Mbps である。

管理用ノードで pickup を動作させ、ディスクイメージを保存した。pickup は、対象のノードをディスクイメージを作成するための専用のディスクレスシステムで起動するよう変更し、Magic Packet Technology[5] による Wake on LAN によりノードを起動するか、SNMP により再起動する。pickup は、このディスクレスシステム上で動作する SpringOS のモジュールと通信し、対象のパーティションをファイルサーバに保存する。

3.2 実験概要

netperf はサーバプログラムである netserver とクライアントプログラム netperf から構成される。利用者は netserver をまえもって起動しておき、netperf から netserver に対してトラフィックを生成することで、その間の帯域を測定する。評価実験用トポロジは図 2 のように構成した。評価実験では図中の netserver と netperf のペア数 n を変更し、実験に必要な時間を計測した。対応するペアは一つの VLAN に所属し、複数のペアが存在する場合は、それぞれ別の VLAN に属するように設定を行った。本実験では一台のノード上では一つの netserver もしくは netperf が動作することとする。また、今後 netserver が動作するノードをサーバノード、netperf が動作するノードをクライアントノードと呼ぶ。実験に利用したノードの情報を表 1 に示す。

表 1: 実験に利用したノードの情報

チップセット	ServerWorks LE
CPU	Pentium 3 1GHz
メモリ	512Mbyte
HDD	IDE 30GB
ネットワーク インターフェース	FastEthernet * 2 GigabitEthernet * 1

実験の管理用ノードには、表 1 と同じ構成のノードを 1 台確保し、SpringOS を導入した。実験用ノードには FreeBSD 5.5 をインストールし、FreeBSD の ports[6] を利用して netperf をインストールした。

3.3 実験のシナリオ

本実験のタイムラインを図 3 に示す。図中では実験用ノード 2 台（サーバノード server[0]、クライアントノード client[0]）と管理用ノード 1 台（master）のタイムラインの概要を示した。実際に実験を行うノードは、サーバノードとクライアントノードであり、管理用ノードは実験用ノードを管理するためのノードである。より大規模な実験は、サーバノードとクライアントノードのペアを増やすことにより実施した。ただし管理用ノードは実験用環境の規模に関わらず一台のみである。図中 *callw* はコマンドのフォアグラウンド実行を *call* はバックグランド実行

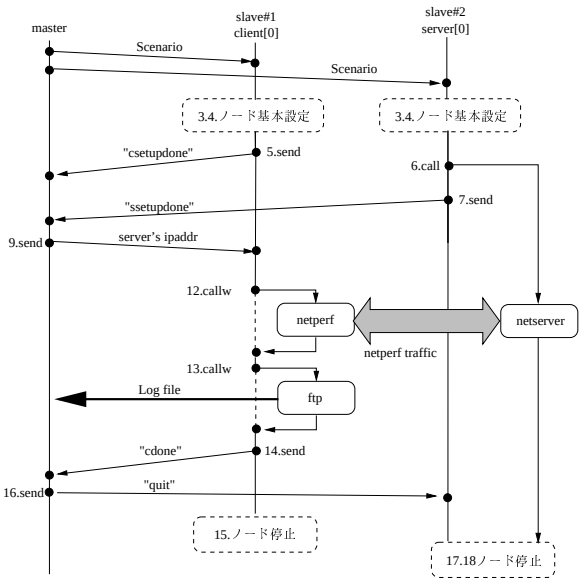

図 3: 評価実験のタイムライン

を示す。ただし、図 3 では、実験の概要を示すため、ノードの初期設定や、疎通確認部分など補助的なイベントは省略している。

以下で、ソフトウェア導入後に、シナリオとして実行される処理を列挙する。

1. 実験用ノードが起動し slave が起動する。
2. master は、各実験用ノードへシナリオを配布すると、クライアントノードからの”csetupdone” とサーバノードからの”ssetupdone” メッセージを待つ。
3. ログファイルに保存される時刻などを同期するため、ntpdate コマンドを実行し管理側ネットワークに接続された NTPd と通信することにより時刻同期を行う。
4. ifconfig を実行し実験ネットワークに接続されたネットワークインターフェースの IP アドレスを設定する。
5. クライアントノードの slave は netperf を実行できる状況であることを master へ通知するため”csetupdone” メッセージを送信する。
6. サーバノードの slave は、ネットワークインターフェースの設定後、netserver をバックグラウンドで実行する。
7. サーバノードは netperf コマンドによる帯域測定の準備が整ったことを master へ通知するため、”ssetupdone” メッセージを送信する。
8. この時点で master が待っていたメッセージが揃う。
9. メッセージがそろったことをトリガとして、master は各クライアントノードへ、対応するサーバノードの IP アドレスを送信する。
10. master は IP アドレスを送信した後、すべてのクライアントノードからの”cdone” メッセージを受け取るまで待機する。
11. サーバノードの IP アドレスを受信したクライアントノードの slave は、サーバノードへの到達性を ping により確認する。
12. その後 netperf を実行し帯域を測定する。これらの実行ログは標準出力に出力されるため、リダイレクト機能を利用してファイルに保存する。
13. netperf 終了後、クライアントノードの slave は、FTP で管理用ノードに保存したログを送信する。
14. クライアントノードの slave は、実験が終了したことを master に知らせるため”cdone” メッセージを master へ送信する。
15. “cdone” メッセージ送信したクライアントノードの slave は shutdown コマンドを実行し、ノードの電源を切る。
16. master は、すべてのクライアントノードから “cdone” メッセージを受け取ると、すべてのサーバノードへ”quit” メッセージを送信する。
17. “quit” メッセージを受け取ったサーバノードは pkill コマンドにより netserver を終了する。
18. サーバノードはメッセージを送信後 shutdown コマンドの実行によりノードの電源を切る。

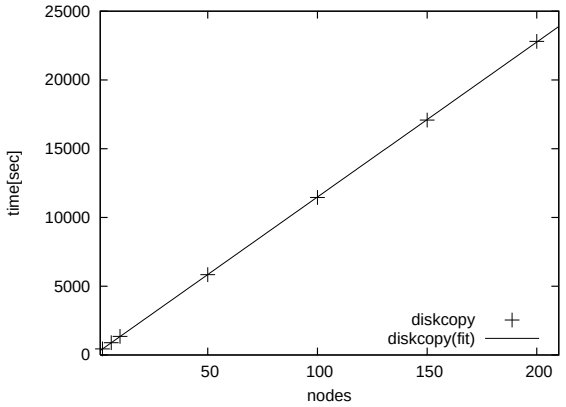

図 4: ノード数とディスクコピーの所要時間

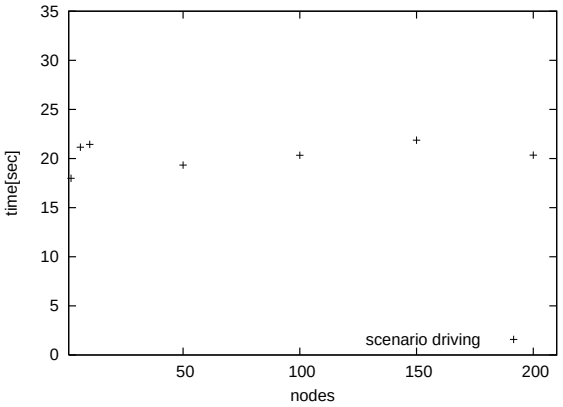

図 6: ノード数とシナリオ実行の所要時間

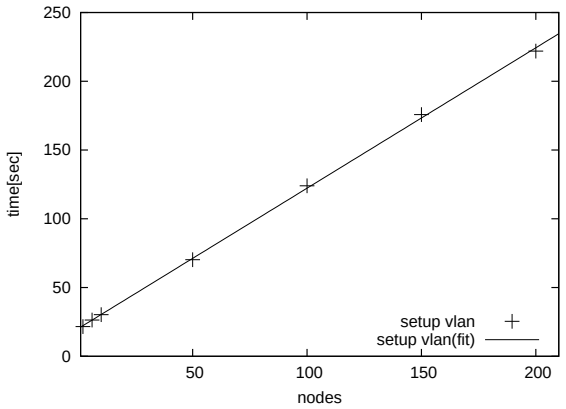

図 5: ノード数と VLAN 設定の所要時間

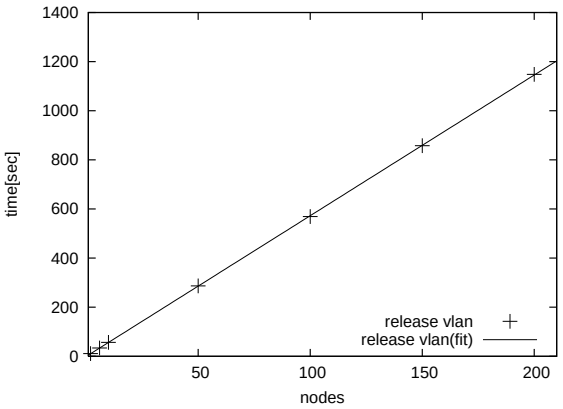

図 7: ノード数と VLAN 設定削除の所要時間

4 実験結果

以下の項目について、計測された結果を示す。

1. ディスクイメージの作成
   - OS とアプリケーションのインストール
   - ディスクイメージの `pickup` による保存
2. 実験の実行
   - 実験用ノードへのディスクイメージの配布
   - 実験用スイッチの VLAN 設定
   - シナリオの実行
   - VLAN 設定の削除
   - 全体

4.1 ディスクイメージの作成

ディスクイメージの作成のため、実験用ノードと同一構成のノード 1 台に手動で FreeBSD 5.5 をインストールした。インストールは Express モードで行い、パーティション設定などはすべてインストーラによる自動設定を利用した。また、利用したパーティションサイズは 4Gbyte で、パッケージは `ports` を含めてすべてインストールした。起動は CD-ROM から行い、パッケージは、ftp3.jp.freebsd.org から FTP で取得した。2 回この作業を試行した結果、所要時間の平均は 1170 秒であった。インストールした FreeBSD への `netperf` のインストールおよび、SpringOS の必要なモジュールとして `slave` と、ノードの再起動や終了を行うための SNMP のサーバプログラムのソースコードのダウンロードおよびコンパイルとインス

表 2: 台数と所要時間（平均値（秒））

ノード数	ディスクコピー	VLAN 設定	シナリオ実行	VLAN 削除	合計
2	442.4	21.6	18.0	11.1	507.9
6	896.5	26.3	21.2	33.9	993.5
10	1352.2	30.2	21.4	56.7	1475.7
50	5843.6	70.0	19.3	282.6	6230.8
100	11456.2	124.0	20.0	569.0	12185.1
150	17079.7	175.7	21.9	858.0	18151.0
200	22806.9	222.0	20.3	1148.0	24213.2

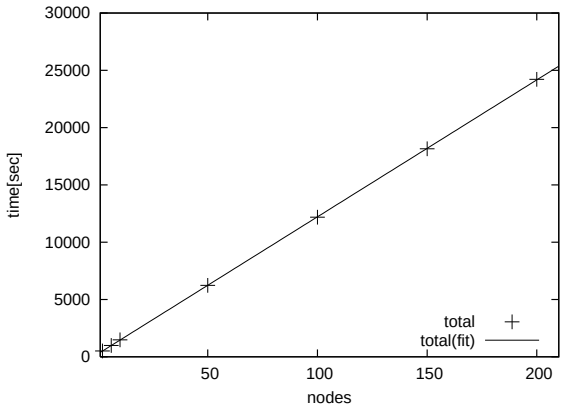

図 8: ノード数と実験全体の所要時間

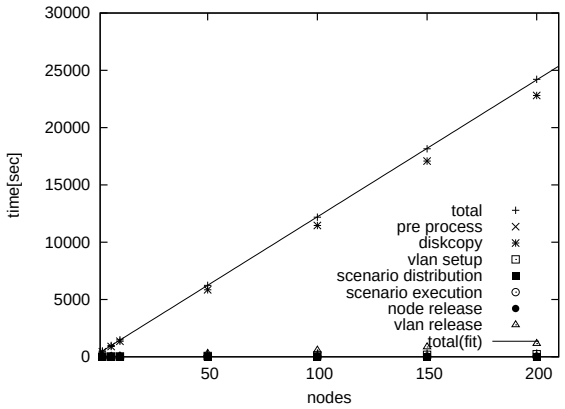

図 9: 実験全体の所要時間

トールは合計で 1 分ほどで終了した。

この後、管理ノードで pickup を動作させ、FreeBSD をインストールしたノードの該当パーティションを保存した。pickup は 4 回試行し、その平均時間は 1284 秒であった。

4.2 計測結果

実験用ノード数を 2、6、10、50、100、150、200（1 ペア〜100 ペア）と変化させ、それぞれの台数での実験の所要時間を 3 回ずつ測定した。それぞれの台数での平均値を表 2 にまとめ、それぞれの段階でのノード数と所要時間を図 4 〜 8 に、合計時間を図 9 に示す。また図 9 には合計時間だけでなく、ディスクコピーの時間もプロットした。最小自乗法を利用した結果、比例関係が認められるグラフには近似直線を示した。合計は表に示した値の総和ではなく、master による設定記述の読み込みと解析や、slave へのシナリオの配布、ノードの予約解放にかかった時間も含めた master の実行時間を示す。

なお、今回の実験ではノードの初期化は行っていない。ただし、ハードディスクに初期状態のディスクイメージを書き込めば良いため、ディスクコピーと同じだけの時間を要すると考えられる。

5 考察

本章では 4 章で示した実験結果の考察を行う。図 4、5、7 からわかるように、ディスクコピー、VLAN 設定、VLAN 削除はノード数と所要時間に比例関係がある。一方、図 6 のシナリオの実行には特に比例関係は見られない。これは、SpringOS がシナリオを並列に実行するためである。

ディスクコピーも同様に平行して行われるが、ディスクイメージが保存されているファイルサーバの性

能および、図 1 のスイッチ F とファイルサーバ間、スイッチ D とスイッチ F 間の帯域がボトルネックとなり、ノード数に応じて所要時間が増加している。

前述のように、ノードへの OS のインストールを手作業で行った場合には、一台につき約 20 分必要である。この場合、十数台までの実験であればインストール終了までに数時間が必要となるが、SpringOS を用いれば 10 台のノードへのディスクイメージの導入は 22 分程度（表 2）で完了できる。ただし、実際にはディスクイメージの元となる OS のインストールとして約 20 分と、ディスクイメージの作成のための pickup の実行に 21 分ほどかかるため、合計で 1 時間強の時間が必要である。したがって、人手で一台ずつインストールした場合の 3 台分の時間で、10 台のインストールが実現できるといえる。また、それ以上の台数では人手でインストールをすることは現実的ではないが、SpringOS を用いれば、図 2 に示したように、200 台で 6 時間強とディスクイメージの作成時間として 40 分強を足しても、7 時間程度で実現できる。台数が多くなればディスクイメージ作成にかかる時間が無視できるため、一台あたり約 2 分で OS とアプリケーションソフトウェアのインストールが可能である。

今回の実験では 1 ペアにつき 1 つの VLAN を用意したため、設定が必要なポート数および、VLAN の作成数はノード数に比例して増加した。したがって、設定時間がノード数に比例することは自然である。VLAN の設定を人手で行う場合は、各ノードが接続されているスイッチのポートの確認など、非常に煩雑な作業が必要である。また、この理由から設定に間違いが生じやすい。SpringOS では、これらの情報をリソースマネージャが保存し、その情報により設定が自動化されるため、正確かつ迅速な設定を行える。

図 6 はシナリオ実行の所要時間を示している。シナリオ実行の所要時間はノード数に関係ないことが見て取れ、100 ペア程度の実験であれば、制御性能が十分であることがわかる。人手により実験を実行した場合には、各ノードでコマンドを入力することになり、これだけの短時間で実験を終了させることは困難である。ただし、本実験では `netperf` による単純な帯域測定を並列に行っているため、このように短時間でシナリオの実行が終了している。より長い実行時間が必要な実験シナリオでは、実行するシナリオに応じた実行時間が必要となる。

また、図 9 からは、ディスクコピーの所要時間が実験所要時間の大半をしめることが見て取れる。ディスクイメージの変更が必要でなければ、シナリオでノードを `shutdown` せず、繰り返し実験に利用することができる。その場合は、利用する実ノードとシナリオ内の論理ノードの対応を記述し、`master` にディスクコピーをさせないように実行する。これにより、ディスクコピーが必要ない、小さなパラメータの変更のみによる再実験などを効率的に行える。

本章では、`master` が動作している時間を所要時間として表現してきた。しかし実際には `master` を実行するだけで、設定記述に記された実験はすべて自動で行われる。したがって、200 台の実験であっても、設定記述を用意し `master` を実行すれば実験実行者は実験が終わるまで、別の作業をすることができる。すなわち、実験の設定記述が用意できれば、実験実行者の実験への拘束時間は非常に小さい。

また、実験結果からは `netperf` は正常に実行されており、実験実行者が記述したシナリオにしたがい正確に実験が実行されていることがわかった。

これらから、StarBED および SpringOS では、以下の 2 点を達成したといえる。

- 人手で構成することが現実的でないような大規模な実験の容易な実行
- 設定記述にしたがった正確な実験駆動単位の構築とシナリオ実行

人手で実験を実行するには時間的に困難であるだけでなく、多数の設定が必要になる実験では、すべての設定を間違いなく行うことは困難である。SpringOS を利用すれば、機械的に多数のノードの設定およびシナリオ実行が行われるため、手動での実行に比べ高い精度と再現性が確保できる。

6 おわりに

本論文では、StarBED 上で SpringOS を利用した実験を行い、各段階に必要な時間を計測した。これらの結果から、StarBED および SpringOS を利用し

た実験の自動実行により、実験の実行者が実験に拘束される時間の大幅な削減、実験精度の向上、実験の再現性の確保など、さまざまな利点があることが確認できた。

本実験の結果からはディスクイメージの導入にかかる時間の比率が非常に大きいことが見て取れる。本実験ではディスクイメージを一台のファイルサーバからすべてのノードへユニキャストで送信しているが、これをマルチキャストで行う方法や、数台の実験用ノードに送信したディスクイメージをさらに別の実験用ノードに配布することでファイルサーバとファイルサーバまでの通信路の負荷を分散させるといった方式を採用することで性能の向上が期待できる。このような点に考慮し、さらに詳細な計測を行うことにより、StarBED の設備と、SpringOS の支援ソフトウェアの性能の向上を図る。

参考文献

[1] Toshiyuki Miyachi, Ken-ichi Chinen and Yoichi Shinoda, “StarBED and SpringOS: Large-scale General Purpose Network Testbed and Supporting Software,” in *International Conference on Performance Evaluation Methodlogies and Tools （Valuetools） 2006*, Oct. 2006.

[2] Toshiyuki Miyachi, Ken-ichi Chinen and Yoichi Shinoda, “Automatic configuration and execution of internet experiments on an actual node-based testbed,” in *International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities（Tridentcom）*, Feb. 2005.

[3] Ken-ichi Chinen, Toshiyuki Miyachi and Yoichi Shinoda, “A rendezvous in network experiment — case study of kuroyuri,” in *International Conference on Testbeds and Research Infrastructures for the Development of Networks and Communities（Tridentcom）*, Mar. 2006.

[4] The Public Netperf Homepage, http://www.netperf.org/.

[5] *Magic Packet Technology*, Advanced Micro Devices, Inc, Nov. 1995.

[6] The FreeBSD Documentation Project, *FreeBSD Handbook Chapter 4 Installing Applications: Packages and Ports*, http://www.freebsd.org/doc/en_US.ISO8859-1/books/handbook/ports.html.