

StarBED: 大規模インターネット 実証実験環境

2006年3月

StarBED プロジェクト

宮地 利幸

知念 賢一

篠田 陽一

概要

従来のインターネットは実験的な側面を非常に強く持っており、実証環境としても利用されていた。しかし、現在のインターネットは社会基盤として利用されており、すでに様々な重要なサービスが運用されている。このようなサービスに影響をおよぼす恐れがあるため、もはや一利用者の判断によりインターネットを実証環境として利用できない。その一方、新たな技術の導入のための動作試験や、インターネット運用技術者の育成を目的として、インターネット上で様々な実験を行いたいという要求は高まってきている。

この矛盾を解決するため、インターネットと分離された疑似インターネット環境が求められている。このような環境を提供する手法の一つであるソフトウェアシミュレータはインターネット上のノードやネットワークの挙動を巨視的にシミュレートするソフトウェアである。ソフトウェアシミュレータを用いれば、様々な規模のネットワークを容易に構築することができ、構築したネットワーク上での各要素の制御も容易であるが、実際にインターネット上で利用される実装を利用できないことが多い。

一方、実ノードを用いて実験トポロジを構築すれば、実際にインターネットに投入される機器やソフトウェアを利用できるため、インターネットの挙動に近い環境を構築できる。しかし、機器の調達、物理的な接続や設定、ノード制御のためのコストがソフトウェアシミュレータを利用した場合と比較して大きい。このため、数台から十数台程度の実ノードを用いた小規模な実証環境が利用されることが多い。しかし、実環境との同一性を考えた場合、実ノードを用いた大規模な実証環境で容易に実験が行えることが理想的である。我々はこの要求を満たす大規模な実ノードによる実証環境である StarBED および、StarBED での実験を支援するためのシステムである SpringOS を実装した。計算を目的とした計算機クラスタは多数存在しているが、StarBED は実験を行うことを目的とした計算機クラスタである。StarBED を利用することで小規模な実証環境上では観測できなかった挙動の観測が可能になる。また規模的に実環境に近い実証環境を構築することができるため、実験の信頼性および実環境との同一性の向上が期待できる。

本論文では、大規模な実証環境の必要性とその実装である StarBED および SpringOS について述べる。

第1章 大規模実証環境

本章では、ソフトウェアシミュレータや小規模な実ノードによる実証環境といった既存の実証環境について述べ、大規模実証環境の必要性を議論する。

1.1 既存の実証環境とその特性

ソフトウェア開発の各段階において、それぞれの環境の適応性は異なる。アイデアの検討など初期段階では、巨視的に新たな技術の性質を確認できるため、ソフトウェアシミュレータは有用である。プロトコルの手順の検証などは、実験対象以外の要素の挙動を隠蔽し、実験遂行者の想定する環境で検証が行えるため、ソフトウェアシミュレータが効果を発揮できる実験である。ある程度ソフトウェアシミュレータでの検証を行った後は、実環境に近い環境での実装の検証を行うことが望ましい。実ノードによる実証環境はこの段階で非常に有用である。これにより、実環境へ導入する実装の正当性や性能評価を行える。

図 1.1 にネットワークアプリケーション開発手順の一例を示す。開発者は、まずアイデアをソフトウェアシミュレータで確認し、ある程度効果があると認められれば、実環境用に実装し、小規模な実ノードによる実証環境で検証を行う。この段階で効果が十分あり、また既存のサービスに影響を与えないことを確認後、実環境に導入する。実環境に導入後、なんらかの問題が発生した場合は、その環境を小規模な実証環境上に再現後、問題に対する対応を行い、修正した実装を実環境に導入するということを繰り返す。しかし、検証用に利用した実ノードによる実証環境と、インターネットのような実環境との差異は、規模性、多様性ともに大きい。

インターネットのような実環境に新たな技術を導入する際には、ノードやその集合であるネットワークのどのような性質が技術に影響するのか、また新たな技術が既存の技術に及ぼす影響の種類や規模の予想は困難である。ソフトウェアシミュレータは実験遂行者が前もって想定した環境で、決められた手順にしたがって正確に動作するが、各要素の動作を十分に予測できていなかった場合は、その結果が実際のインターネット上での挙動と異なる恐れがある。実ノードを用いたネットワークであれば、インターネットと同様に、それぞれのノード上のプロセスが自律的に動作し、それが集合することで、実験前には想定できなかった状況が生まれる場合もある。

しかしこれまでは様々なコストの問題から、十数台から数十台規模以上の大規模な実証環境を構築するのは、コストなどの観点から困難であった。しかし、このような小規模な実ノードによる実証環境は、実環境と規模性、多様性においての乖離が大きい。このため小規模な実証環境での実験を行ったあと、実環境にアプリケーションを導入した際に、実験段階では予測できなかった問題が発生することがあった。このような問題は、小規模な実ノード実証環境と実環境の中間に位置する大規模実証環境で実験を行うことにより、事前に発見できる可能性がある。また、ソフトウェア開発のより早い段階で実ノード環境での実験を行えることも利点の一つである。実環境で用いられる実装を早い段階から評価できるため、実環境での新技術の特性をより詳細に認識でき、その結果を開発にフィードバックできる。

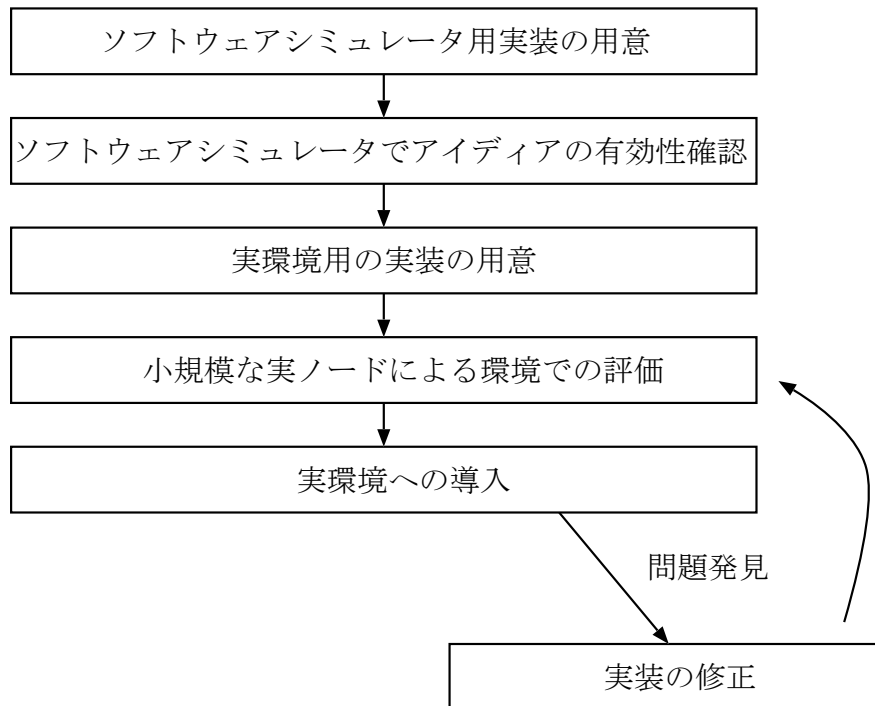


図 1.1: ネットワークアプリケーション開発ステップの一例

我々は、数百台の実ノードを一カ所に集めた環境を構築し、それらのノードを用いた実証環境の構築および、実験遂行のための支援ソフトウェアを充実させることを考えた。これにより、実ノードによる実証環境の実環境との同一性と、ソフトウェアシミュレータの実験のしやすさという長所をあわせ持つ実証環境を構築できる。このような多数の実験設備と支援ソフトウェアが存在する実証環境を利用することにより、既存の実証環境構築技術よりも大規模な実験トポロジを柔軟に構築できる。また、支援ソフトウェアによりある程度ノードの設定を自動化できるため、多様性を向上させることも可能である。本論文では、大規模な実験設備に実験トポロジの構築と実験の遂行を自動化するソフトウェアが用意された環境を大規模実証環境と呼ぶ。

1.2 大規模実証環境の設計

実ノードによる実証環境の構築および、実験の遂行コストが大きいことはすでに述べた。実証環境の規模が大きくなるほど、ノードの用意や物理接続に関するコストは大きくなる。我々はこれらのコストを軽減するため、多数の実験用実ノードをもつ施設を構築することで、実証環境を構築する度に実ノードを調達するコストの低減を図った。この施設を複数の実験遂行者で時分割および空間分割により共有利用することで、その経済的および人的な管理コストを低減できる。

大規模実証環境には、非常に多くのノードが存在するため、そのノードを一台ずつ管理し、実験トポロジの構築を行うことは困難である。この問題を解決するため、外部からの設定入力により、自動的にノード間接続を変更できる機器を利用する。このような機器には、VLAN や ATM により仮想的にトポロジを変更できるスイッチや、物理的に接続を変更できるスイッチが利用できる。

これまでで挙げた手法により、実ノードの用意およびトポロジ構築のためのコストは軽減でき

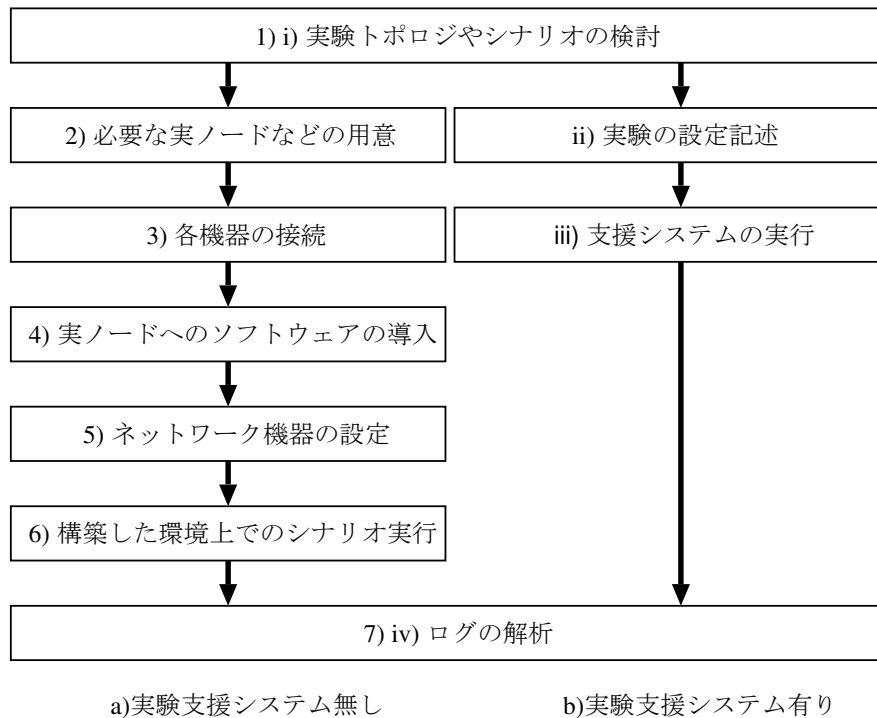


図 1.2: 実験の遂行手順

る。しかし、ノードへのソフトウェアの導入や実験シナリオの遂行にも、大きな人的および時間的リソースが必要となる。また人為的ミスにより、実験精度の低下を招く可能性は高い。これを解決するため、実験遂行者の意図する実験トポロジの自動的生成および、実ノードへ自動的に OS やアプリケーションを導入する支援システムを用意し、実験遂行者は実験の遂行の大部分を支援システムにまかせることで、人為的な実験精度の低下を防ぐ。

本章では、このような環境を想定した実験の遂行手順について考察し、実験設備に必要なハードウェア要件および、支援ソフトウェア要件について議論する。

1.3 実ノード環境での実験の遂行手順

一般的な実ノードによる実証環境での実験の遂行手順について、支援システム無しの場合を図 1.2 a)(アラビア数字) に、支援システム有りの場合の手順を図 1.2 b)(ローマ数字) に示す。

それぞれの手順について以下に示す。

- 1) 実験トポロジやシナリオの検討 実験の目的により適切なネットワークトポロジや各ノードの性能要件およびに導入するソフトウェアを検討し、さらにその環境上で実行されるべきシナリオを検討・決定する。
- 2) 必要な実ノードなどの用意 検討の結果から必要な実ノードやネットワーク機器を用意する。
- 3) 各機器の物理的な接続 用意したネットワーク機器を物理的に接続する。

- 4) 実ノードへのソフトウェアの導入 実験用ノードへ必要な OS やアプリケーションを導入し、必要な設定を行う。
- 5) スイッチなどネットワーク機器の設定 必要であれば、スイッチやネットワーク機器の設定を行う。
- 6) 構築した環境上でのシナリオ実行 構築された環境上で実験シナリオを遂行し、実験データを収集する。
- 7) ログの解析 得られた実験データを解析する。

2の手順についてはすでに提案した、前もって多数のノードを用意しておくという方法で解決できる。支援ソフトウェアは何らかの形で実験実行者による実験設定を、入力として受け取り、それに従い実験が遂行する。この時の実験手順は、図 1.2 b) に示した通りであり、実験遂行者は実験の大部分を支援システムにまかせることで、実験による拘束時間を短縮することができる。これにより、実験遂行者は行う実験の検討およびその結果の解析に注力できる。また、実験は前もって記述された手順にしたがって機械的に遂行されるため、人為的なミスの低減および、再現性の向上にも効果がある。

支援ソフトウェアが行う処理は、実験用に用意されているリソースから、実験トポロジを構成するために必要なリソースの選択および獲得、実験トポロジの生成、実験シナリオの実行である。

1.4 実験設備への要件

まずは実証環境として動作させるための基本的な機能として、実験設備への要件をあげる。

管理用ネットワークの分離 実験用のネットワークと、管理用のネットワークを分離する。これにより実験の通信と制御用通信の相互の影響を防ぐ。

リンク特性の模倣 リンクの遅延やパケット損失、ディレイなどのリンク特性を生成するための機能が用意されていることが好ましい。専用の機材を利用する方法や、広帯域の対外線を用意し、実環境の特性を取り入れる方法がある。

ノードへのコンソール操作 ネットワークが切断された際に、実験ノードへ接続し、問題点の特定や復旧を行うため、各ノードのコンソールを一カ所から、容易に操作する機能が必要である。

自動的な実験トポロジの自動構築 すでに述べたように、自動的にネットワークの構築を行える機材が必要である。

1.5 支援ソフトウェアへの要件

実験支援ソフトウェアへの要件を以下にまとめる。

実験遂行者による設定の認識・解析 実験遂行者による実験ノード設定および実験トポロジについての設定を認識・解析する。この結果から他の機能と連携し、実験トポロジを構築し、実験シナリオを遂行する。

リソースの状態・属性管理 利用可能な実ノードや実験トポロジを構築するためのリソースの状態およびそれらのリソースの属性を管理しておき、その情報をもとに実験遂行者へ実験に必要なリソースを割り当てる。属性にはノードのネットワークインターフェースの種類や数、アーキテクチャや標準でインストールされている OS などがある。実験トポロジを構築するためのリソースとして、VLAN 番号などがある。複数の実験遂行者による空間分割利用のため、排他処理が必要である。

実験ノードの自動設定 実ノードに OS やアプリケーションを導入する機能である。これにより各実ノードは実験用ノードとして振る舞う。

実験トポロジの自動構築 自動的に実験トポロジを構築する機能である。設定記述にしたがって、ネットワーク機器の設定を行う。

シナリオに従った実験の自動遂行 設定記述にしたがって、実験を遂行するための機能である。実験遂行者による各ノードのプログラムの起動の人的・時間的コストを削減できる。また人為的な実験ミスを低減する効果もある。

実験ノードの状態管理/表示 各実験ノードがどのような状態にあるかを理解しやすい形式で実験遂行者に知らせる。

実験ログ収集 実験終了後、各実験ノードからログを収集する。実験の規模が大きくなるほど、ログ収集のための人的および時間的なコストが高くなる。

リンク特性の模倣 ハードウェアの項ですでに述べたが、Dummynet[1] や NIST Net[2] などのソフトウェアでの実現も可能である。

また、何らかの方法で実ノードの電源管理を行える手法が必要である。IPMI[3] や Magic Packet Technology[4] による Wake on LAN などを利用して、ハードウェアレベルで実ノードの電源を ON/OFF できる機器はさまざまなベンダによって発表されている。またソフトウェア的に電源の管理を行うためには SNMP などを利用することができる。

第2章 StarBED

我々は、前章の内容をふまえて大規模実証環境の一実装として StarBED[5] を提案した。StarBED は通信・放送機構により北陸 IT 研究開発支援センターとして 2002 年に開所された。本章では StarBED について述べる。

2.1 StarBED の構成概要

StarBED は 512 台の実ノードと、VLAN および ATM の VC/VP を用いて仮想的に変更することで、実験トポロジを構築する。StarBED の概念的なトポロジを図 2.1 に示す。

StarBED において実験トポロジは、VLAN および ATM の VP/VC を用いて設定可能である。実験トポロジの設定や、実ノードへの設定は管理用ネットワークを通して行われる。管理用ネットワークは、管理用トラフィックの実験への影響を防ぐため実験用ネットワークと分離されており、各実験用ノードは管理用と実験用の 2 つのネットワークに接続されている。実験を支援するためのファイルサーバや DHCP サーバ、支援システムの一部のモジュールは管理ネットワークに接続されたノードで動作する。WIDE Project の 10Gbps のネットワークおよび、Japan Gigabit Network(JGN) の 10Gbps のネットワークに接続されており、必要であればこれらのネットワークを通じてインターネットへ接続できる。このネットワークを利用して別のサイトへの接続や、実トラフィックの導入に利用できる。各ノードのコンソールは Raritan 社 [6] の製品により、一台の端末から操作できるため、ネットワークからの制御が不可能になった場合にも対応できる。また、さまざまな実験を行うため、仮想トラフィック生成装置や、パケットスニファなどのハードウェアも用意されている。実験遂行者の持ち込みハードウェア専用のラックも用意され、柔軟に実験に対応できる。

StarBED には 512 台の実験専用の PC と、それらを接続するネットワーク機器で構成されている。これらの PC のネットワーク接続を VLAN および ATM の VC/VP を用いて仮想的に変更することで、実験トポロジを構築する。StarBED の概念的なトポロジを図 2.1 に示す。

2.2 仮想機械の利用

我々の想定する実証環境は基本的に多数の計算機を集め、1 計算機を実験トポロジ上の 1 ノードとして動作させることで大規模な実験トポロジを構築する。存在する計算機では実現できない規模の実験トポロジを構築するために、その都度、新たにノードを用意することは、経済的費用、設置するための空間、そして計算機の保守費用などさまざまなコストが必要となるため現実的ではない。この解決策として、1 台の計算機を仮想的に多重化することで、さらに大規模な実験トポロジを構築する手法がある。すでに用意されている計算機を仮想的に多重化すれば、新たな計算機を用意するためのコストは発生せず、計算機の保守費用は増加しないため、新たな計算機を用意する場合よりも、さまざまな点でコストが小さい。

我々が想定する実証環境は実環境用の実装を大規模な実験トポロジ上で検証することが目的で

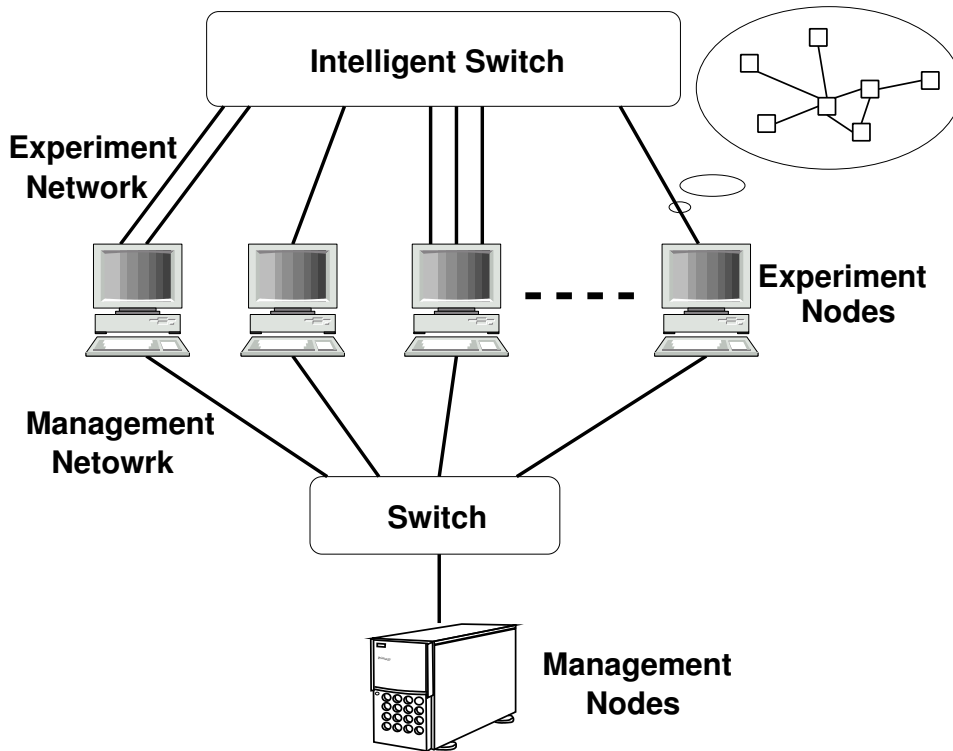


図 2.1: StarBED の概念的トポロジ

ある。したがって、実験遂行者が要求するソフトウェアが変更を加えることなく動作することが非常に重要である。汎用的な OS が動作しない場合は、実験遂行者が意図するアプリケーションの動作に支障をきたす場合や、実験遂行者が変更を加えた OS を利用できないなど、問題が生じる場合がある。仮想ノード実現技術の一つである仮想機械は、ハードウェアレベルで計算機を模倣する。したがって、汎用的な OS が変更なしに動作し、ソフトウェアレベルでは実環境と同一の挙動を示す仮想ノードを利用できる。ただし、仮想機械の性質や性能に影響を受けるため、性能検証や、実ノードのデバイスドライバなどといった計算機自体の機能に関する検証には適さない。

一般的に、仮想機械より高い抽象度で仮想ノードを実現する技術では、専用の OS を利用し仮想ノードの動作速度を向上させているが、OS の模倣を行わずプロセス等のレベルでの多重化を可能にする。このような技術を利用すれば、更に大規模な実験トポロジを構築できるが、実環境との同一性は低下するため、実験遂行者は実験の性質を十分に吟味し、仮想ノードを利用できるかどうかを決定する必要がある。

StarBED にはノード多重化のため、仮想機械の実装の一つである、米 VMware 社の VMware[7] を導入し、数千台規模の実験を可能とした。

第3章 SpringOS

SpringOS は StarBED で利用されている実験支援システムの総称であり、個別の機能を持つモジュール群により構成されている。各種設定を記述したファイルを用意するだけで、実験遂行者の意図する実験が、SpringOS により自動的に遂行される。SpringOS の主な機能を以下にあげる。

- 実験遂行者による設定の記述の認識・解析
- リソースの状態および属性管理
- ノードへの OS の導入および各種設定
- スイッチへの VLAN 設定による実験トポロジ構築
- シナリオを管理するシナリオマスタと、シナリオを実行するシナリオスレーブとが協調することによるシナリオ実行

電源管理については Wake on LAN および SNMP を用いて実現している。また、シナリオ遂行機構を用い、Dummynet や NIST Net を起動することによるリンク特性の模倣を、ファイル転送用プログラムを起動することで実験ログの収集を実現している。

3.1 ノードへのソフトウェアの導入

ノードのソフトウェア導入には、前もって実験遂行者が作成したディスクイメージをノードのハードディスクに書き込むことで行う方法と、ディスクレスシステムで各ノードを動作させる方法に対応している。ディスクイメージを生成する支援モジュールも用意しており、一台のノードに必要なシステムを構築すれば、その環境を多数のノードに複製できる。ディスクレスシステムとして起動する場合は、起動が非常に短時間で行える。ディスクレスシステムを構築する方法には、ディスク部分を TFTP を利用し取得後メモリ上におく方法および、NFS を利用する方法が利用できる。

TFTP を用いたディスクレスシステムでは、ノードは起動する際に TFTP 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 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、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ため、TFTP で転送できる最大のファイルサイズは約 32MB 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場合はこの手法は利用できない。

NFS を利用した場合はネットワークを経由し、ディスクにアクセスを行うため、頻りにディスクにアクセスする場合は、この負荷が実験結果に影響をおよぼす恐れがある。ただしログの記録程度であれば、ログ保存用スペースを実験ノード上に確保することで回避できる。実験ノードへのソフトウェア導入の詳細については、[8] にまとめられている。

3.2 シナリオの自動遂行

実験のシナリオも設定ファイルに記述されることを前提とする。また、シナリオが記述された設定ファイルを読みこみ、シナリオ全体を管理する要素をシナリオマスタと呼ぶ。

シナリオマスタは実験遂行者の記述したファイルを読み込み、実験前に各実験ノード用のシナリオを配布する。各実験ノードが別のノードの挙動をトリガとしてイベントやプログラムを実行するといったノード間同期が必要な場合はシナリオマスタを通じ、メッセージパッシングを用いて行う。OS やアプリケーションの導入および設定が済んだノードでは、実際にノード上でコマンドを実行するシナリオスレブが起動する。シナリオマスタは定期的にノードへ接続を試み、シナリオスレブが起動後、接続が成功すると、シナリオスレブはシナリオマスタから実行すべきシナリオを受け取りシナリオ遂行を開始する。

シナリオスレブはシナリオに記述されたプログラムの起動や、ノードの同期のためシナリオマスタへメッセージの送信および、シナリオマスタからのメッセージを受信するまで、シナリオ遂行を停止するといった処理を行う。また、シナリオマスタは各ノード同期のためのシナリオを実行する。シナリオの実行についての詳細は [9][10] にまとめられている。

3.3 設定記述

一般的に一つの実験で同様の設定のノードが多数利用されることが多いため、我々が実装した設定記述ではオブジェクト指向的なクラス概念を導入した。これにより同一の設定の多数のノードを容易に宣言することが可能である。

設定記述は設定部とシナリオ部に別れており、設定部ではノードのブート方法や、ディスクを利用する場合はディスクイメージの指定、ディスクイメージを書込むハードディスクのパーティション、必要なネットワークインターフェースの種類、ノードで実行されるシナリオなどが記述される。

シナリオ部にはノードのクラス設定部に記述されるノードシナリオと、ノードシナリオの協調のためにシナリオマスタが実行するグローバルシナリオがある。シナリオ設定部は `senario` 宣言で始まるブロックに記述される。ノード間協調は実験ノード上で動作するシナリオスレブと、管理用ノード上で動作するシナリオマスタのメッセージ交換によって実現される。ノードシナリオは、基本的にノードで実行されるコマンドのリストであり、他のノードとの協調が必要な場合のみ、メッセージ交換のための記述がされる。グローバルシナリオはノードシナリオの協調のために利用されるため、メッセージ交換の制御が主な内容となる。

これらの設定記述の例を図 3.1、図 3.2、図 3.3 に示す。この例はノードを 2 台用意し、一台で `netserver` を起動し、もう一台のノードで `netserver` が起動しているノードに対して `netperf` を実行するだけという簡単なものである。図 3.1 はノードクラスの宣言とインスタンスの生成、図 3.2 はネットワークに関する設定、そして図 3.3 はグローバルシナリオの例である。

図 3.1 では、`svclass` という名前のノードクラスを宣言している。このクラスのノードのディスクイメージである `img.gz` は、FTP を利用して `172.16.1.1` から取得され、パーティション 2 に書込まれる。また Ethernet のネットワークインターフェースが 1 つだけ利用される。シナリオ部には、ノード起動後 `netperf` のサーバプログラムである、`netserver` を起動し、シナリオマスタに `"serverstarted"` というメッセージを送信、その後シナリオマスタから `"quit"` というメッセージを受け取ると `netserver` のプロセスを終了する内容のシナリオが記述されている。最終行で、`server` という名前のインスタンス群を 1 台生成している。

図 3.2 では、`ethclass` という名前のネットワーククラスを宣言している。IP アドレスのレンジを

```

nodeclass svclass {
    partition 2
    ostype "FreeBSD"
    diskimage \\\
        "ftp://usr:pas@172.16.1.1/img.gz"
    netif media fastethernet
    scenario {
        wake "/sim/netserver" \\\
            "/sim/netserver"
        send "serverstarted"
        recv msg
        msgswitch msg {
            "quit" {
                wakewait "/usr/bin/killall" \\\
                    "killall" "netserver"
                exit
            }
        }
    }
}
nodeset server class svclass num 1

```

図 3.1: ノード定義

ここで指定することで、参加したネットワークインターフェースに自動的にアドレスを設定できる。netset 宣言で ethnet という ethclass のインスタンスを生成している。attach でノードのインターフェースをこのネットワークに参加させることで、ネットワークを形成する。

図 3.3 はグローバルシナリオの一例である。このシナリオでは、開始後、server[0] からの”serverstarted”というメッセージと client[0] からの”clisetupdone”というメッセージを待つ。この2つのメッセージを受信すると、client[0] へ server[0] の IP アドレスをメッセージとして送信する。その後 client[0] から”cdone”というメッセージを待ち、受信すると、server[0] へ”quit”というメッセージを送信し終了する。

SpringOS は実ノードおよび VMware による仮想機械を制御可能であり、ある実験に必要な数の実ノードが確保できない場合や、実環境との同一性を吟味し、仮想機械で十分要求が満たせる場合は、仮想ノードを利用し実験トポロジを大規模化できる。

SpringOS の設計および詳細については、[11] にまとめられている。

```
netclass ethclass {
    media fastethernet
    ipaddrange "192.168.3.0/24"
}
netset ethnet class ethclass num 1
attach server.netif[0] ethnet
```

図 3.2: ネットワーク定義

```
scenario {
    sync {
        msgmatch server[0] "serverstarted"
        msgmatch client[0] "clisetupdone"
    }
    send client[0] \\  
        haddr(server[0].netif[0].ipaddr)
    sync {
        msgmatch client[0] "cdone"
    }
    send server[0] "quit"
    exit
}
```

図 3.3: グローバルシナリオ定義

第4章 StarBED で行われた実験の考察

StarBED は多くの研究に利用され、さまざまな成果を残している。我々は、これまでの運用の結果から実証環境に求められる性質について議論した。本節では、StarBED で行われた実験について分析する。

[12] では、階層型 IP トレースバック機構を提案、実装し、小規模な実ノードによる環境での動作検証が行われており、より実環境に近い大規模な実験トポロジ上でのデータの取得のため StarBED の利用を検討している。実環境用の実装を用いる必要があったため、ソフトウェアシミュレータの利用は困難である。また、多くの AS 間の通信を模倣した大規模な実験トポロジが必要であるため、研究室レベルでの実験トポロジ構築は困難である。運用環境を意識しての検証であるためこれは、構成要素の挙動の同一性と、環境の規模の問題から規模追従性が重要であった。

[13] では、Multi-player Online Games(MOGs) への peer-to-peer 型通信の導入を提案している。小規模な環境での動作検証後、実用化に向けさらに大規模な環境として StarBED での動作検証を行っている。実環境用のソフトウェア利用と約 300 ノードを利用した実験トポロジの構築は、ソフトウェアシミュレータ、研究室レベルの実ノードを用いた環境での実現は困難であり、測定する対象が実時間であったことから、構成要素の挙動の同一性と規模追従性と実時間性が重要であった。

[14] は、マルチキャストを用いたグループ通信が ISP のバックボーンに対しどのような影響を与えるのかを検証するため、StarBED における仮想ノードを含め 1000 ノードを越える実験トポロジを構築し実験を行った際の工夫や知見をまとめたものである。この時、仮想ノードはグループ通信のユーザによるトラフィック生成部分に利用し、コアネットワーク部分は実ノードにより構成された。また、この際には、奈良先端科学技術大学院大学へ JGN を介して接続しトラフィックを迂回させ、実ネットワークの遅延などの特性を実証環境に取り入れた。この実験には、規模追従性および挙動の同一性が重要であった。

また、これ以外にも、さまざまなスイッチのマルチキャストトラフィックの転送能力の検証や、TCP の挙動の観察、大規模なトポロジ上でのトラフィック解析、有名ベンダのルータとオープンソースで開発されているソフトウェアルータの性能比較などが StarBED を用いて行われた。

これらの実験では、実環境に利用するための実装を、実ノードによる大規模かつ現実的な実験トポロジ上で検証したいという要求が大きい。これはソフトウェアシミュレータ、研究室レベルの実ノード環境では実現できない。また、実時間でのデータ測定が必要である場合も多い。当然、実験を容易に行えるための仕組みはすべての実験で重要である。しかし、これまで行われてきた実験では、実験を容易に行えるという点よりも実環境との同一性が重要視された。

また、VPN や VLAN などの技術を用い、StarBED の各ノードをインターネットのさまざまな位置に出現させ、その位置のトラフィック情報を計測することにより、さまざまなプロトコルの系としての動作の観察を行うような実験も計画されている。

第5章 関連研究

X-Born[15] や Planetlab[16] は、分散したサイトの実ノードを接続し、大規模な実証環境を構築するものである。我々が提案している環境は、一つのサイトに多くの計算機を用意することにより実現するものであるため、比較対象とはならない。以下では、Netbed と ModelNet について考察する。

5.1 Emulab/Netbed

Netbed[17] は、分散システムと実ノードによる環境およびシミュレータの統合環境であり、豊富な実験管理機能をもつ。変更可能なパッチパネルをソフトウェアにより制御し、物理的な結線を変更しトポロジを変更できる。この機能と VLAN をあわせて利用することで柔軟な実験トポロジを構築できる。ns-2 を実ネットワークに接続できるように拡張した nse[18] を利用し、ソフトウェアシミュレータと実ノードによる環境を接続しており、ns-2 を利用している部分では、前述の通り実環境と同一の実装は扱えない。

ある程度の規模の PC クラスタをもつサイトを接続することで、大規模な環境を構築しており、実験トポロジがさまざまなサイトに分割されることがある。これにより、サイト間の接続の問題が実験に影響をおよぼす可能性がある。また、何らかの問題が発生した場合に、実験対象とサイト間の接続部分のどちらに問題があったのかを切り分けるのは困難である。

StarBED は一般的な計算機を一カ所に集めることで、管理コストを低減し、また、別ネットワークから完全に分離された大規模な環境を提供することができる点が Netbed とは異なる。

Netbed の利点として実験ネットワーク中に実ネットワークの遅延を導入できることが挙げられるが、一カ所に集中した施設であっても、遅延などをエミュレートできる。むしろ、分散配置による制御の遅延や帯域の制限などが問題になると予想できるが、開発中の環境をインターネットに接続することになるため、その脆弱性も問題である。

Netbed のシステムは、実験遂行者からのリクエストを受付け、施設が空き次第実験を遂行する。実験は一括処理で遂行されるため、実験を行いその状況を判断して次の実験を遂行したい場合や、教育を目的とした場合などの実験トポロジをある状態まで自動的に構築し、その後は実験遂行者の手動により操作したい場合には不向きである。

5.2 ModelNet

ModelNet[19] は実ノードと同様のソフトウェアが動作する Virtual Node を一台の実ノード上で多重化し大規模なトポロジを構築し、さらにコアノードでこれらの Virtual Node 間のリンク特性を変更するためのソフトウェアである。

Virtual Node を利用しているため各ノードの物理的な特性の検証や、物理的な特性に関連するソフトウェアなどは利用できず、性能測定用途にも利用は不可能である。また Virtual Node 自体が実

ノードとどの程度の同一性をもっているかの検証がなされていない。

コアネットワークで、多数のパスに対しリンク特性をエミュレートするため、規模耐性の問題がある。

ModelNet はソフトウェアであり、ハードウェアとソフトウェアの両面から検討がなされている StarBED とは異なる。また、複数の実験遂行者による共有利用については考慮されていない。

第6章 まとめ

インターネットのような大規模な実環境に、新たな技術の実装を導入する前には、できるだけインターネットに近い環境で検証を行うことが望ましい。これまでの小規模な実ノードによる実証環境で行った実験結果は、場合によっては、インターネットのような大規模で多様な実環境での挙動と大きく異なる場合がある。また、小規模な環境では観測し得ない計測結果を、大規模な環境では捕らえることが可能である。

我々は、このような問題を解決するため、大規模実証環境を提案し、実際に実装するにあたり様々な検討を行った。大規模実証環境を利用すれば、実ノードによる大規模な実験トポロジの構築および、シナリオの遂行を低コストかつ容易に実行できる。これにより、実ノードを利用することによる実環境の同一性を持ち、ある程度大規模な実証環境が実現可能となる。

本論文では、大規模実証環境の一実装である StarBED の設計と、支援システムである SpringOS の動作について述べた。その上で、これまでに StarBED で行われた実験について考察し、その他の方法では実現困難な実験例から、我々の提案した実証環境の有効性を示した。

関連図書

- [1] L. Rizzo. Dummynet: a simple approach to the evaluation of network protocols. *ACM Computer Communication Review*, Vol. 27, No. 1, pp. 31–41, 1997.
- [2] NIST Internetworking Technology Group. *NIST Net network emulation package*. <http://www-x.antd.nist.gov/nistnet/>.
- [3] Intel Corporation. *IPMI v2.0 specifications Document Revision 1.0*, February 2004.
- [4] Advanced Micro Devices, Inc. *Magic Packet Technology*, November 1995.
- [5] The StarBED Project. <http://www.starbed.org/>.
- [6] Raritan, Inc. <http://www.raritan.com/>.
- [7] VMware. <http://www.vmware.com/>.
- [8] 三角真, 宮地利幸, 知念賢一, 篠田陽一. 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. 情報処理学会研究報告書 DPS-116, pp. 95–100, January 2004.
- [9] 宮地利幸, 知念賢一, 篠田陽一. StarBEDにおける自動実験駆動機構. 第6回インターネットテクノロジーワークショップ (WIT2004).
- [10] 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)*, March 2006.
- [11] Toshiyuki Miyachi and 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)*, February 2005.
- [12] Masafumi OE and Youki Kadobayashi. An implementation and verification of a hierachial trace-back. In *Wiley Electronics and Communications in JAPAN, Part 1 Vol.87, No.11*, pp. 49–56, Nov 2004.
- [13] Takuji Iimura, Hiroaki Hazeyama, and Youki Kadobayashi. Zoned federation of game servers: a peer-to-peer approach to scalable multi-player online games. In *NetGames*, 2004.
- [14] Eiichi Muramoto, Takahiro Yoneda, Atsushi Nakamura, Makoto Misumi, Toshiyuki Miyachi, and Yoichi Shinoda. Report on a method of simulating multicast group communication on the internet. Towards peta-bit ultra networks, September 2003.

- [15] X-Born. <http://www.isi.edu/x-bone/>.
- [16] Planetlab website. <http://www.planet-lab.org/>.
- [17] Brian White, Jay Lepreau, Leigh Stoller, Robert Ricci, Shashi Guruprasad, Mac Newbold, Mike Hibler, Chad Barb, and Abhijeet Joglekar. An integrated experimental environment for distributed systems and networks. pp. 255–270, Boston, MA, December 2002. USENIXASSOC.
- [18] The VINT Project. *Network Emulation with the NS Simulator*. <http://www.isi.edu/nsnam/ns/ns-emulation.html>.
- [19] Amin Vahdat, Ken Yocum, Kevin Walsh, Priya Mahadevan, Dejan Kostic, Jeff Chase, and David Becker. Scalability and accuracy in a large-scale network emulator. In *The Fifth Symposium on Operating System Design and Implementation (OSDI02)*, December 2002.