修 士 論 文

大規模ネットワークの実証環境向け抽象化技術に関する研究

北陸先端科学技術大学院大学
情報科学研究科情報科学専攻

明石 邦夫

2010 年 3 月

修 士 論 文

大規模ネットワークの実証環境向け抽象化技術に関する研究

指導教官　篠田陽一 教授

審査委員主査　篠田陽一 教授
審査委員　日々野靖 教授
審査委員　敷田幹文 准教授

北陸先端科学技術大学院大学
情報科学研究科情報科学専攻

0810002 明石 邦夫

提出年月: 2010 年 2 月

概 要

インターネット上では，物理的な要因によって遅延や帯域幅の制限，パケットロスなどといったインターネットの特性が存在する．アプリケーションや製品の検証をインターネットの特性を考慮した実験ネットワークで行うことによって，より品質を向上させることができる．遅延や帯域などを模倣するソフトウェアとして，ネットワーク特性エミュレータが存在する．ネットワーク特性エミュレータは，ノード上で遅延の生成や，帯域の制限などを行うことができるソフトウェアである．ネットワーク特性エミュレータが，遅延の生成や帯域の制限などを行う際，ノードが持つCPUやメモリなどのリソースを消費する．しかし，ノードが持つリソースを考慮したインターネットの特性を模倣する実験ネットワークの構築手法は提案されていない．本研究の目的は，実証環境上に遅延や帯域などといったインターネットの特性を模倣した大規模な実験ネットワークの構築技術の提案である．

本研究では，ノードが持つリソースを効率的に利用し，可能な限り大規模な実験ネットワークを構築する手法を目的とする．ネットワーク特性エミュレータでは，インターネットの特性の模倣を多重化できる点に着目し，1台のノードで複数のノード間の特性を模倣する．1台のノードが模倣可能な限界を超えてしまうと正しくインターネットの特性を模倣できない．そのため，ノード間に設置するインターネットの特性を模倣するために必要なリソースの算出を行う．リソースとは，1台のノードが持つCPUやメモリ，ネットワークインターフェースの処理能力である．複数のインターネットの特性を模倣する際に，これらのリソースを超えないようネットワーク特性エミュレータの配置を行う．

もう一つのアプローチとして，検証を行うためにインターネットの特性すべてを模倣する必要はない点に着目した．本研究では，検証を行うために，インターネットで観測できる遅延や帯域，パケットロス，IPルーティングをインターネットの特性として定義する．そして，検証を行う実験ネットワークにインターネットの特性を導入する際，それぞれの特性を抽象化し，取捨選択する．インターネットの特性を抽象化することによって，実験者は検証内容に応じて，必要な特性のみを考慮した検証を行うことが可能となる．本研究では，インターネットの特性を抽象化し，多種多様な検証方法に対応できる実験ネットワーク構築システムの実現を目指す．

本研究では，実際にテストベッドにおいて，インターネットの特性を模倣した実験ネットワークの構築実験を行った．また，インターネットの特性を模倣する関連研究を複数取り上げ，本提案システムとの比較を行い，それぞれのメリット，デメリットを考察した．本提案システムにおけるインターネットの特性を模倣した実験ネットワークでは，ノードが持つリソースを限界まで利用する．そのため，限られたリソースしか扱えない環境においても，扱える範囲のリソースを効率的に用いて実験ネットワークを構築することを可能とした．

目 次

第1章　はじめに

1.1　背景

インターネットは，重要な社会基盤として利用されている．その上では Web サーバや動画配信など，様々なサービスが動作している．これに伴い，インターネットを利用することを前提とした技術が数多く開発・利用されている．新たに開発された技術や製品を十分な検証を行わずにインターネットへ導入することは，インターネットに重大な影響を及ぼす可能性があるため避けるべきである．そのため，新たなネットワーク技術を開発しインターネットへ導入する前に，十分な検証を行い製品の質を向上させる必要がある．

ネットワーク技術の検証は，利用を想定している環境であるインターネットの利用が理想的である．しかし，インターネットは，重要な社会基盤として多数のユーザやサービスに利用されている．十分な検証が行われていない技術をインターネットへ導入すると他のユーザやサービスに影響が出る可能性がある．そのため，インターネットを検証の場として利用することは好ましくない．インターネットを利用している他のユーザやサービスに影響を考慮せず，検証を行う方法が必要である．

一つの解決策として，インターネットから隔離されたテストベッドを利用する方法が挙げられる．インターネットから隔離したテストベッド上で実験ネットワークを構築することにより，他のユーザやサービスに与える影響を考慮しなくても良いメリットがある．しかし，インターネットから隔離されたテストベッド上で遅延や帯域を考慮した検証を行うには，実験者が遅延や帯域の模倣を行う必要がある．

遅延や帯域の模倣を行うためには，専用のソフトウェアを利用することで解決できる．遅延や帯域の模倣には，ソフトウェアを動作させるノードの CPU やメモリなどのリソースが利用される．そのため，ノードが持つリソースを超えて模倣を行った場合，遅延や帯域の模倣が正しく行われない．実験者は，ノードのリソースと遅延や帯域の模倣に必要なリソースを考慮して，ソフトウェアの設定を行う必要がある．

インターネットから隔離された実験環境では，遅延や帯域などを自動的に模倣するシステムは提供されていない．そのため，実験者が遅延生成や，帯域制限などの模倣に必要なリソースを考慮しつつ，実験ネットワークを手作業で構築する必要がある．

1.2 目的

限られたリソースの中で可能な限り大規模な実験ネットワークを構築するには，利用可能なリソースを効率良く使用する必要がある．

本研究では，検証に必要と考えられる遅延や帯域などの特性を抽象化し，検証に必要とされる特性のみ模倣する環境構築手法を提案する．提案手法では，実験者が検証項目毎に要求する特性を保持しつつ，実験ネットワーク上に要求される遅延や帯域を模倣する．これにより,実験ネットワークの構築に必要なリソースを抑えることが可能となる．また，遅延や帯域の模倣に必要なリソースを計算し，1 ノードが持つリソースの限界を超えない範囲で模倣を行う．この手法により，限られたリソースの中で可能な限り大規模な実験ネットワークを構築することが可能となる．このような実験ネットワークは実験者から見た場合に，抽象化は行われているが，要求した特性を満たしている．したがって，検証を行うのに問題のないネットワークを構築することが可能である．この技術により，実験者は利用可能なリソースを意識せずに，大規模な実験ネットワークを構築することが可能となる．

1.3 本研究の構成

2 章では，ネットワークソフトウェアの検証方法と求められる特性について述べる．3 章では，検証のための関連研究について述べる．4 章では，抽象化した大規模ネットワークの構築手法を提案し設計する．5 章では，実装した提案システムについて述べる．6 章では，本研究の評価及び関連研究との比較を行う．7 章では、本研究のまとめと今後の課題について述べる．

第2章　ネットワークソフトウェアの検証環境の要求

2.1　ネットワークソフトウェアの検証環境

インターネットを利用したアプリケーションや製品は，日々増えている．インターネットには，背景トラフィックや遅延，帯域，パケットロスなど物理的な機器の処理能力の限界による影響が強く出る．このような，インターネットを利用する際に影響を受ける性質のことをインターネットの特性と呼ぶ．アプリケーションや製品を検証する際には，インターネットの特性を考慮して検証を行う必要がある．作成したアプリケーションをインターネットへ導入した際，導入したアプリケーションのみが動作している状態は存在しない．そのため，新たに作られたアプリケーションや製品を検証する項目として，インターネットの特性を考慮した検証が必要である．

アプリケーションや製品を検証する際，最終的な利用環境となるインターネットを検証の場として利用することが理想的である．インターネットを利用して検証を行えば，インターネットへ導入した際に受けるであろう影響をそのまま受けて検証をすることができる．そのため，実験結果として限りなく現実に即した結果を得ることができる．

しかし，インターネットには実験を行う以外のユーザやサービスが動作している．実験者は，検証を行うアプリケーションや製品が他のユーザやサービスにどのような影響を与えるかあらかじめ考慮しなければならない．また，実験者は，他のユーザやサービス，ネットワーク機器を自由に設定，操作することはできない．実際にインターネットを用いた検証では，突発的なバーストや，急激な遅延の増加などが起こる可能性がある．検証の段階で，インターネットの特性の急激な変化は考慮すべきである．しかし，検証の際にアプリケーションがこれらの影響を必ず受けることができるとは限らない．仮に，希望通りの影響をうけることが出来たとしても，その状況を再現することは不可能であるという問題がある．

そのため，実験環境を実験者自らが構築し，インターネットから隔離された環境で検証する手法が良く用いられる．このような環境を本研究では実験環境と呼ぶ．実験環境ではインターネットから隔離されているため，インターネットを利用している他のユーザやサービスへの影響を考慮しなくてもよいというメリットがある．実験環境では実環境のための実装がそのまま利用できる．そのため，新たな技術の仕様に記述されていない挙動やバグまでを含め，アプリケーション，製品自身の詳細な検証結果が期待できる．

しかし，実験環境を構築するためには，実験ネットワークを構築するために必要なノー

ドを物理的に配置し，それぞれを接続し，必要であればネットワーク機器の設定を行う必要がある．また，ノードを用意することによる経済的・空間的コストおよび，ノードの接続，ネットワーク設定のための人的・時間的コストが大きい．さらに，構築するネットワークの規模が大きくなるにつれ，それぞれのコストも大きくなるため現実的ではない．そして，インターネットから隔離された環境での検証では，インターネットの特性による影響を受けることができない．そのため，検証するアプリケーションや製品が，他のユーザやサービスにどのような影響を及ぼすのか検証できない問題がある．

上記の手法における問題から，検証を行う場としてテストベッドが存在する．テストベッドとは，多数の実験用ノードやネットワーク機器が実験設備として提供されている．また，多数の実験用ノードやネットワーク機器を制御するための支援ソフトウェアが提供されている．テストベッドを用いることで，実験環境で問題となる経済的コストを抑えることができる．テストベッドは，特定の用途に特化したものや汎用的に用いることができるよう作られたものなど様々な種類が存在する．

様々な用途に用いられるテストベッドの中でも，ネットワーク技術の検証のために利用できるテストベッドを本研究では実証環境と呼ぶ．実証環境では，一般的に多数のノードを実験ノードとして利用でき，実験者は提供されているノードを利用してアプリケーションや製品の検証を行うことが可能である．インターネットの特性が存在するかという観点では，実証環境ではインターネットを利用しているものとインターネットから隔離されたものに分けることができる．

インターネットを利用している実証環境では，他のユーザやサービスが利用している場を検証環境として用いる．そのため，インターネットを検証の場として利用した場合と同じく，他のユーザやサービスからの影響をそのまま受けることができる．インターネットを利用している実証環境では，実験環境を利用するメリットである経済的・時間的・人的コストの削減は期待できる．その一方，検証を行うアプリケーションや製品が他のユーザやサービスに与える影響をあらかじめ考慮しなければならないデメリットは残る．

インターネットから隔離された実証環境では，検証するアプリケーションのみが存在する検証環境を構築することができる．メリットとして，経済的・時間的・人的コストの削減と，他のユーザやサービスに与える影響を考慮しなくても良いという点が挙げられる．しかし，インターネットの特性を考慮した検証を行いたい場合，実証環境としてインターネットの特性を模倣する機構は提供されていない場合が多い．そのため，検証する際に，インターネットの特性を模倣した実験ネットワークを実験者が構築する必要がある．手作業でインターネットの特性を模倣した実験ネットワークを構築するには，人的・時間的コストが大きくなってしまうデメリットがある．

2.2 インターネットの特性を模倣した実験ネットワークの必要性

インターネットから隔離された実験環境では，他のユーザやサービスが存在せず，物理的な距離も短いため，インターネットの特性が存在しないネットワークが構築される．しかし，インターネットでは常に他のユーザやサービスが動作しており，インターネットの特性が存在しない状態で利用できることは少ない．例えば，国を跨いだ通信では非常に長い距離を通信するため，遅延は大きくなる．また，中継するネットワーク機器の回線が他のユーザによって圧迫されていた場合には，利用可能な帯域は狭くなる．このような環境を想定した実験ネットワークを構築した上で検証を行うことで，より品質の向上が見込まれる．

実験者が，検証を行いたい環境を想定する場合，実験環境自体が小規模であればソフトウェアやハードウェアを用いることで遅延・帯域を模倣した実験ネットワークの構築は可能である．しかし，規模が大きくなるにつれて，実験ネットワークの構築にかかる人的・時間的・物理的コストが高くなり現実的に困難になる．そのため，大規模な実験設備において，どのように遅延・帯域を考慮した実験ネットワークを構築するかが問題となる．

大規模な実験設備においても遅延・帯域などのインターネットの特性を模倣した実験ネットワークが必要である．また，実験設備の規模に関わらず，容易に構築可能なインターネットの特性を模倣した実験ネットワークの構築手法が必要である．

2.3 模倣するインターネットの特性

インターネットの特性を模倣した実験ネットワークを構築する際に，実際のインターネットで観測できるインターネットの特性をすべて模倣する必要はない．例えば，実験者が非常に大きな遅延が発生する環境を想定しているのであれば，帯域の模倣は行わずに遅延の生成だけ行えば良い．逆に，トラフィック転送レートの制御の検証を想定しているのであれば，遅延は考慮せずに帯域の模倣だけ行えば検証には十分である．本研究では，インターネットの特性を模倣したネットワークの構築に必要とされる要素を定義する．これらの要素により，物理的な距離感，利用可能な帯域，不安定な通信などを模倣することが可能となる．また，これらを組み合わせることによって，検証を行うために必要なインターネットの特性のみを模倣した実験環境を構築することが可能となる．本項では，定義する要因がアプリケーションにどのような影響を及ぼすかを解説する．

2.3.1 背景トラフィック

インターネットは，序章で述べたように重要な社会基盤として利用されている．その中には多数のユーザやサービスが利用しているため，検証対象が生成する以外のトラフィックが流れている．

インターネットに導入した際に，これらの背景トラフィックが存在する状態でも動作せねばならない．背景トラフィックについては梅木ら [1] の研究によって，実証環境内に他のユーザやサービスを模倣した背景トラフィックの生成に成功している．

2.3.2 伝送遅延

ネットワークにおいても，遅延は発生する．これは，高速ネットワークや高性能な機器によって解決できるものではない．遅延時間を短くすることはできても，なくすことはできない．

ネットワークを利用するアプリケーションが，ファイル転送のようなバッチ形式の処理ばかりであれば，遅延はあまり気にならない．しかし，IP 電話やビデオチャットなどに代表される双方向でのインタラクションが必要なアプリケーションでは，遅延に関する考慮が必要になる．

2.3.3 遅延揺らぎ (ジッタ)

遅延を引き起こすものではないが，非常に重要な要素に伝送遅延の変動，ジッタ (ゆらぎ) がある．すべての通信に対して常に同じ遅延時間が発生するのであれば，それを予測して対処することは可能である．しかし，遅延時間は必ずしも一定ではない．特にインターネットの場合，どこを経由してパケットが転送されるかがわからないため，パケットの到着時間を予測するのが非常に困難である．音声アプリケーションでは，遅延揺らぎが大きくなると会話が聞き取りづらくなる．映像アプリケーションでは，遅延揺らぎが大きくなると映像を正しく描画できずノイズが入ってしまう．このように，音声・映像を扱うアプリケーションでは，遅延揺らぎが原因となり正しくデータを送信できない現象が発生する．そのため，遅延揺らぎによるアプリケーションの挙動はインターネットへ導入する前に検証を行っておくべきである．

2.3.4 帯域幅

インターネットを構成しているノードやネットワーク機器及びそれらを接続しているケーブルには流れる情報量の限界が存在する．ノードやネットワーク機器を経由して情報 (データ) を転送する際の転送レートを帯域幅と呼ぶ．

この帯域幅はノードやネットワーク機器が持つネットワークインターフェイスの最大転送速度が物理的な限界となる．つまり，ノード間の通信時には一番狭い帯域幅を持つ部分がボトルネックとなる．例えば，図 2.1 のように，Host-A，B は 1Gbps の帯域を利用可能だとしても，Router-A，B 間で利用可能な帯域幅が 100Mbps であった場合，Host-A，B 間で利用可能な最大帯域幅は 100Mbps となる．また，他のユーザやサービスも同じ回線を利用している場合は，複数のデータが 1 本の回線帯域を共有するため，1 つのデータ

が利用できる帯域幅は狭くなる．例えば，図 2.2 では，Host-A，B 間で利用可能な帯域幅は 100Mbps であるが，Host-C が Host-D や Host-E と通信している場合，Host-A，B 間で利用可能な帯域幅は 100Mbps よりも狭くなる．

音声通信や映像配信を行うアプリケーションの検証を行う場合，適切な転送レートで通信することが求められる．帯域幅の模倣を行うことにより，実験者は極端に狭い帯域しか利用出来ない環境や，帯域幅の変動が激しい環境で検証することができる．

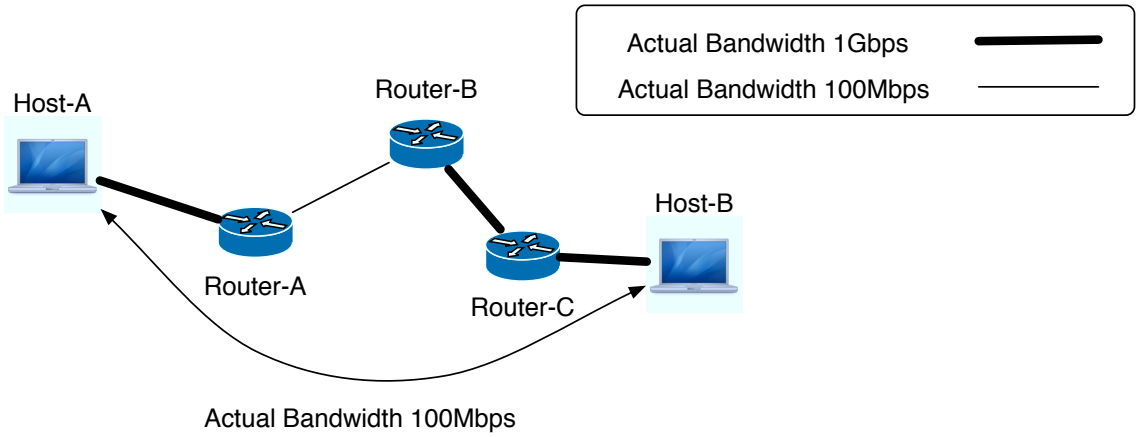

図 2.1: 帯域のボトルネック

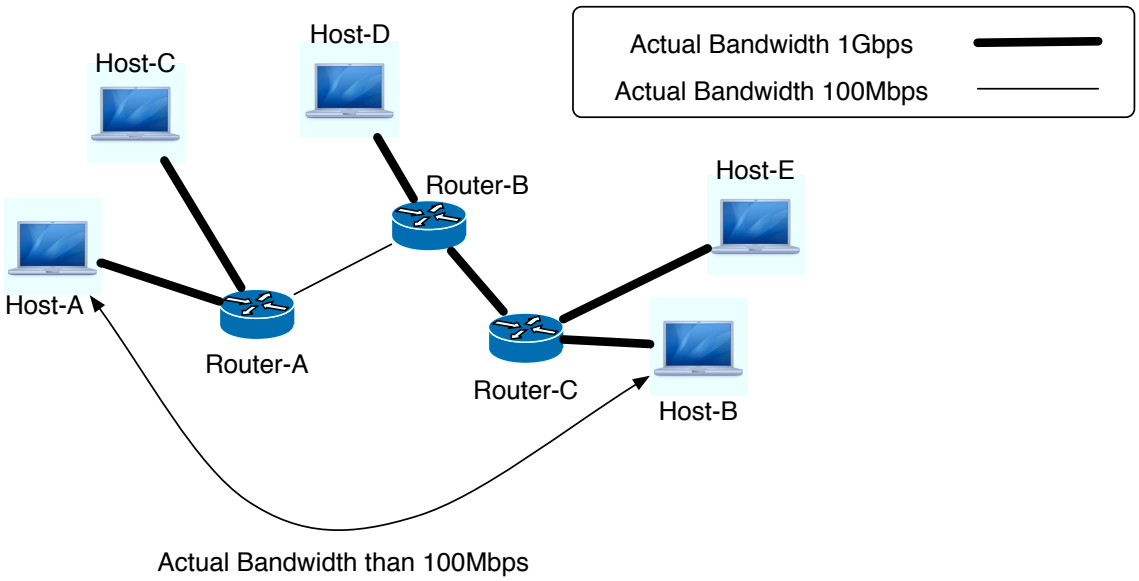

図 2.2: 他のユーザやサービスによる帯域のボトルネック

2.3.5 パケットロス

IP ネットワークでは，最善努力 (ベストエフォート) 型とよばれる通信形態が一般に用いられている．最善努力型通信では，全体の通信が少ないときはネットワークの伝送能力を余すことなく利用できるという利点を持つ．しかし，通信量が増加して回線やネットワーク機器，サーバーの処理能力を超える場合などに，データ (パケット) を廃棄してもよいことになっている．このデータの廃棄をパケットロスと呼ぶ．

Web やメールなど多くのインターネットアプリケーションでパケットロスが発生した際には，パケットを再送してデータを回復する手法が用いられている．しかし、多数の受信端末を相手とするマルチキャストやリアルタイム通信が要求される音声アプリケーションなどでは，送信側やネットワークの負担，リアルタイム性を考えると、個々の受信端末の受信状況に応じて再送を行うことは現実的ではない．リアルタイム通信を行うアプリケーションであれば，パケットロスがどの程度発生しているのかの判断や，パケットロスが一定値以上になった場合に，どのような処理を行うのかなどの制御が必要である．パケットロス発生時の制御が正しく行われない場合，データが正しく送信されないことを意味するので，制御が正しく行われるかの検証は必要である．

2.3.6 IP ルーティング

インターネットでは，ノード間に多数のルータやスイッチといったネットワーク機器が存在する．これらのネットワーク機器は Boarder Gateway Protocol(BGP) [2] や Open Shortest Path First(OSPF) [3] といったプロトコルを用いてネットワーク上の経路を作成している．ルータが故障や，誤った設定がされた時などにインターネットの経路が変化する．これは，ノード間の到達性を確保するために冗長性を持たせているためや，誤った設定によって経路が変更してしまうためである．インターネット上での経路が変更した場合，当然ながらノード間の通信時に経由するネットワーク機器は変化する．そのため，ノード間の伝送遅延や帯域幅も変化する．

2.4 提案

本研究では，これまで述べてきたインターネットの特性を実験ネットワーク上に模倣する手法を提案する．

本提案では，実験ネットワークに遅延や帯域などのすべての特性を完全に模倣する必要はない点に着目する．. 検証を行うために必要な特性のみを模倣した実験ネットワークを構築すれば，検証を行うには十分である．そこで，インターネットの特性を抽象化し，実験者が必要な特性のみを模倣した実験ネットワークを構築する．

これにより，実証環境で利用可能なリソースの中で，可能な限り大規模な実験ネットワークを構築することが可能となる．

第3章　関連研究

3.1　検証のために用いられるテストベッド

本節では，テストベッドの中でもネットワーク技術の検証のために用いられるStarBED [4], Netbed [5]，PlanetLab [6]，およびGARITについて述べる．

3.1.1　PlanetLab

PlanetLabは，米国のIT企業や世界60以上の大学が中心となって2003年に立ち上げられたテストベッドである．PlanetLabの目的は，インターネットを利用したアプリケーションやサービスのための，オープンで拡張性があるグローバルなネットワーク・テストベッドを構築することである．参加条件は，規定されているスペックを超えたノードを提供することである．この条件を満たしていれば，参加者全員のノードを利用することが可能である．2010年現在，PlanetLabには500近い組織が参加しており，ノード総数は1000台を超えている．

実験者は，実験トポロジとしてオーバーレイネットワークを構築し，実験を行う．PlanetLab上のノードはPLノードと呼ばれ，専用のソフトウェアが動作する必要がある．専用ソフトウェアはLinux系OSのRPMパッケージで提供されているため，PlanetLabで実験可能なOSはLinuxのディストリビューションの中でもRPMを扱えるOSに限定される．

また，実験ネットワークの中にインターネットが存在することになるため，インターネットの特性を導入できるメリットが存在するが，問題発生時に原因の切り分けが難しいというデメリットも存在する．

3.1.2　GARIT

GARITは奈良先端科学技術大学院大学に設置されているテストベッドである．GARITに設置されているノードの性能は均一でなく，これらのノード群を制御するためAnyBED [7]が提供されている．AnyBEDは，インターネットのASレベルのトポロジをテストベッドを用いてエミュレート可能である．ユースケースとして，トレースバックや経路制御等の研究，インターネットオペレーションの教育などに用いられている．AnyBEDは，L2スイッチを用いて実験トポロジを構築するソフトウェアであり，GARITに特化したソフト

ウェアではない．そのため，手元の PC クラスタと StarBED のようなハードウェア構成 (ネットワークインタフェイスの数など) が異なるクラスタでも同じトポロジで容易に実験可能である．

また，AnyBED はツール自体がポータブルに設計されている．一度 AnyBed 用マスターサーバの環境を構築しておけば，サーバを持ち込むことにより，様々な PC クラスタ上で実験可能である．

3.1.3 Netbed

Netbed は ns-2 [8] と実機を利用した実験設備である．Netbed では実験者がソフトウェアを用いてネットワークの物理的・論理的トポロジを変更することが可能である．Netbed では実験者の要請を受けた実験を、実験機器が空き次第行う．このとき，割り当てられた利用時間を過ぎた実験環境を保存し，次の利用時に実験環境の設定を復元する．ただし、Netbed は独自のインターフェースで実験環境の構築と実行を行うため，Netbed の規格に沿って構築した実験環境のみを対象としている．

Netbed 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 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．

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，教育を目的し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．

3.1.4 StarBED

StarBED は，北陸リサーチセンターで提供されているテストベッドである．Netbed や PlanetLab では，遠隔地に設置されているノードの制御が困難である場合が多い．またそ

れぞれを接続するネットワーク機器は実験者の管理下にない．そのため，ネットワーク機器の情報の収集や，トラブルが発生した際の問題切り分け及び修正が困難である．そこで宮地ら [9] は，ノードを 1 箇所の施設に集め，ノードを自由に設定することにより，柔軟な実験トポロジの構築が可能と考えた．また，Netbed のような環境では一括処理で実験を行えるため，効率的に実験を行うことが可能であるが，その一方，実験結果から実験内容を修正し再度実験を行うといった繰り返し実験の実行は困難である．このような問題から，宮地らは汎用的なネットワーク実験に利用できる StarBED を設計・実装した．2010 年現在，StarBED には 1000 台を超えるノードとそれぞれのノードを接続しているネットワーク機器から構成されている．実験者は提供されているノードの一部および論理ネットワークを構築するための VLAN 番号を借りて実験を行う．また，実験を行う際には実験者が持参したノードや検証したい実機を持ち込み，ネットワーク機器に接続して実験を行うこともできる．StarBED のノードは，仕様に応じてグループと呼ばれる単位で区切られており，それぞれのグループのノードの性能は同一である．

StarBED では，数多くのノードを利用することにより大規模な実験ネットワークを構築し，検証を行うことが可能であるしかし，大規模な実験ネットワーク環境で実験者が 1 つ 1 つのノードを手作業で操作するのは困難である．そのため，StarBED には実験支援ソフトウェアとして SpringOS が存在する．SpringOS は，実験を支援するためのソフトウェアの集合体であり，ノードの起動，終了を含めた管理，スイッチの VLAN 設定，実験シナリオの実行を行うことが可能である．また，SpringOS の実験シナリオの実行機能を使うことにより，実験内容を修正しながらの繰り返し実験を行うことが可能である．SpringOS の詳細については第 3.1.5 章で解説する．

3.1.5 SpringOS

Experiment Resource Manager(ERM)

StarBED に存在する実験用の資源の管理・割り当てを行う．StarBED に存在する資源とは，ノードや VLAN ID などになる．StarBED で資源の割り当てを行う場合，ネットワークインターフェイスの数や種類，ノードの性能などを ERM に指定し，ERM は実験者が使用することのできる範囲で要求を満たす資源を検索，割り当てを行う．ERM は資源の割り当てを行った後，他の実験者に同じ資源を割り当てないよう排他制御を行う．

Node Initiator(NI)

実証環境上で多数のノードを用いて実験を行う場合，実験の準備に時間がかかる．そこで実験に使用するノード 1 台のみ OS やソフトウェアのインストールを行い，そのノードのイメージを取得した後，他のノードへ配布する機能として NI が存在する．NI は実験用ノード上で動作し，実験用ノードへのソフトウェア導入を行う．NI は各ノードのハードディスクに書き込みを行うため，専用のディスクレスシステム上で起動する．NI 起動後，

ENCD から指示されたディスクイメージをファイルサーバから取得し，ハードディスクに書き込みを行う．

Experiment Node Configuration Driver(ENCD)

ENCD は実験実行の核となるモジュールである．ENCD が実験者による設定ファイルを読み込み，他のモジュールに対して指示を送ることで実験を実行する．ENCD はある一連の動作をさせるための役割により別々に用意されている．例えば，実験を実行するための ENCD は kuroyuri master として実装されている．また，ディスクイメージの作成・配布を行う際にも，他のモジュールを指揮する ENCD が必要となる．これはディスクイメージの作成として pickup，配布として wipeout が実装されている．pickup と wipeout は基本的には前述の NI と組として利用され，NI と通信することでその役割を果たす．

Facility Node Configuration Pilot(FNCP)

StarBED では複数の実験駆動単位が同時に生成される．ある実験用ノードへディスクイメージを導入するために NI が起動した際に，NI は通信すべき ENCD と通信する必要があるが，ENCD の IP アドレスを固定することはできないため，何らかの方法で NI に対象の ENCD のアドレスを指定する必要がある．これを実現する方法として，ENCD から NI に対して通信する方法と，NI が何らかの方法で通信すべき ENCD の IP アドレスを調べる方法がある．ENCD から NI に通信する方法では，NI が起動したタイミングをはかりにくく，効率が悪い場合があるため，SpringOS では後者の方法を採用している．FNCP は StarBED 上で一台のみ動作しており，IP アドレスは固定である．ENCD は自分の IP アドレスと，管理しているノードの管理側ネットワークの固定 IP アドレスを FNCP に登録する．NI は起動した際に FNCP に通信を行い，通信すべき ENCD を確認し通信を開始する．

Directory Manipulator(DMAN)

StarBED では，実験用ノードは PXE でブートローダーを取得し，起動方法を決定している．各ノードが取得するブートローダーに関する設定は，DHCPd の設定として記述されている．そのため，設定を変更した場合に全てのノードに対して影響がでる可能性がある．そこで，SpringOS では DHCPd の設定に記述するブートローダーのファイル名は固定し，そのファイルをシンボリックリンクとして，シンボリックリンクがさすリンク先のファイルを変更することで，変更の影響範囲を単一のノードのみとしている．DMAN は，要求に応じてこのシンボリックリンクの設定を変更する．

Wake on Lan(WoL) Agent

StarBED では実ノードの起動手法の一つとして Wake on LAN [10] を使用している．Wake on LAN は，対象のノードが存在するセグメントで Magic Packet をブロードキャストする．Magic Packet を受け取った対象ノードのネットワーク・アダプタがノードの電源投入を行う．Wake on LAN では，Magic Packet をブロードキャストするため，実験ノードが複数のセグメントに分離している場合に，Magic Packet を中継するための機能が必要となる．WoL Agent は，実験用ノードが動作している全てのセグメントに接続されたノードで動作している．ノードを起動する際には，管理用ノードが WoL Agent に起動要求を行い，WoL Agent が Magic Packet を必要なセグメントに送出する．

SWMG(SWitch ManaGer)

StarBED では Cisco や Foundry といった様々な企業のネットワーク機器が動作している．実験者は様々なネットワーク機器に接続されたノードを用いて実験を行うが，他の実験者も同様に実験を行っている．そのため，他の実験者が生成しているトラフィックと混合してしまう．この現象を避けるため StarBED では，VLAN を用いて L2 レベルでセグメントを分離している．しかし，実験者が使用する VLAN ID はその時々によって変化する．また，1 つの実験の中で多数の VLAN ID を用いて多数のセグメントを作成した上で実験を行いたいと言う要望もある．実験者はその際に，利用可能な VLAN ID の範囲でネットワーク機器に VLAN の設定を行う必要がある．

SWMG は様々な企業が提供しているネットワーク機器の設定方法の違いを吸収し，実験者に同じ使い方で VLAN の設定を反映させる．また，ERM と通信を行い実験者が利用できる範囲の VLAN ID とノードが接続されているネットワーク機器のポートのみ設定反映を許可することを行っている．

3.2 ネットワーク特性エミュレータ

ネットワークの模倣を行う方法は大きく分けて 2 つあることを 3 章で示した．その中で，ソフトウェアによる模倣は実証環境にあるノードを利用できることから，本稿でもソフトウェアによる模倣を用いる．

ソフトウェアによる模倣は，様々なソフトウェアが提供されており，本章では，その中でも自由に利用可能な Linux，BSD 系 OS で利用できるものについて解説を行う．

3.2.1 NISTNet

NISTNet [11] は Linux で提供されているネットワーク特性エミュレータである．NISTNet は Linux 対応のカーネルモジュールの拡張である．OSI 参照モデルにおけるデータリ

ンク層と IP 層の境界で動作する．ネットワークの遅延，帯域制限，パケットロス，パケット重複などをエミュレートすることができる．NISTNet の開発は Linux2.6.14 で終了している．

3.2.2 dummynet

dummynet [12] は FreeBSD，または MacOSX(Tiger 以降) で提供されているネットワーク特性エミュレータである．dummynet のインターフェースは ipfw のユーティリティとして実装されている．現在の実装では，パケットの選別は ipfw プログラムのパイプルールにより行われる．dummynet のパイプは、帯域幅、遅延、キューサイズ、損失率を設定することができる．これらの設定は ipfw プログラムで行う．パイプには，1 から 65534 までの番号がつけられ，パケットは ipfw の設定によっては複数のパイプを介して送出する事が可能である．dummynet は基本的に IP レベルで動作するが，ブリッジ拡張機能を有効にすることにより，ブリッジされるパケットをパイプを介して送出することができる．

3.2.3 NetEm

NetEm [13] は Linux で提供されているネットワーク特性エミュレータである．機能としては，ネットワークの遅延やパケットロス率，パケット再送の模倣をおこなうことができる．Linux のパッケージとして NetEm が提供されているわけではなく，Linux のパケットスケジューラがエミュレートを行う．エミュレートする遅延やパケットロス率の値は，iproute2 [14] で実装されている tc コマンドより操作する．また，NetEm はデバイスのインターフェースキュー (IFQ) を利用して動作するため，基本的に送信時のパケットに適用される．この IFQ は Qdisc(Queue Disipline) と呼ばれ，実際のキューを処理するために使われる．NetEm は，この Qdisc にパケットを格納し，遅延生成，帯域制限の処理を行う．その後，Qdisc からパケットを取り出し，デバイスから送信する．

NetEm 自体は帯域制限の機能を持っていないが，TokenBucketFilter(TBF) や，Class-BasedQueueing(CBQ) などを用いることによって，帯域制限を実現している．

NetEm は，デバイスの IFQ を用いてエミュレートを行うが，基本的に IP 層でパケットフォワーディングを行うことを前提に作られている．そのため，NetEm が動作しているノードがネットワークアドレスが違うものとなる．NetEm では dummynet と同様にブリッジされるパケットに対しても遅延生成・帯域制限をかける機能も提供されている．ブリッジ機能を有効にする場合には，Linux 内でブリッジインターフェースを作成し，そこに所属するインターフェースに NetEm のコンフィグを設定することによってブリッジされるパケットにも適用することが可能である．

NetEm 自体は帯域制限の機能を持っていないため，外部機能を用いることによって帯域制限を実現している．帯域制限を行う方法は様々な方法が提供されており，これを以下に述べる．

- Token Bucket Filter(TBF)
  トークンバケツフィルタ (Token Bucket Filter:TBF) は単純な Qdisc で，管理者が設定した速度を越えない範囲で到着パケットを通す．ただし短い時間の突発的なものなら，この値を越えることを許す可能性はある．

  トークンバケツフィルタは非常に正確で，ネットワークとプロセッサへの負荷も少ない．インターフェースの速度を落とすという機能のみ模倣するのであればトークンバケツフィルタが向いている．

  トークンバケツフィルタの実装はバケツと呼ばれるバッファである．これは定期的にトークンと呼ばれる仮想的な情報の断片によって，一定の割合 (トークン速度) で満たされていく．バケツの最も重要なパラメータはサイズであり，保持できるトークンの数を意味する．

  バケツに入った各トークンは，それぞれひとつの受信データパケットをデータキューから拾い，そしてバケツからは削除される．この 2 つの流れ (トークンとデータ) からなるアルゴリズムには，3 つのシナリオが考えられる．

  1. トークンと同じ割合で，トークンバケツフィルタにデータが到着する．
     この場合各受信パケットは，それぞれ対応するトークンがあるので，遅延することなしにキューを通過する．

  2. トークンの速度よりも遅い割合で，トークンバケツフィルタにデータが到着する．
     キューに入った受信データパケットの出力に応じて削除されるトークンは一部分のみなので，トークンはバケツサイズ一杯にまで溜まっていく．使用されなかったトークンは，突発的なバーストが起こった場合に利用でき，トークンの標準流入速度を越えたデータが送信できる．

  3. トークンの速度よりも大きい割合で，トークンバケツフィルタにデータが到着する
     この場合，バケツのトークンはすぐに空になってしまい，トークンバケツフィルタはしばらくの間入力を絞る．これは「過負荷状態 (overlimit situation)」と呼ばれる．パケットがそのまま入り続けてくる場合には，パケットは破棄され始める．

  後ろの 2 シナリオがとても重要であり，このフィルタを通過するデータのバンド幅を管理者が制御できることを意味している．トークンが溜まっていくと，制限を越えたデータバーストも，短いものならロスなしに通過できるが，過負荷状態が続くとパケットはだんだん遅延していき，ついには破棄される．実際の実装では，トー

クンが対応しているのはバイトであり，パケットではない．トークンバケツフィルタには様々なパラメータが存在する．以下にそれぞれのパラメータとその仕様と解説する．

limit または latency limit はトークン待ち状態でキューに入れるバイト数の制限値である．これは latency(パケットがトークンバケツフィルタに留まれる時間の最大値) を設定することによっても指定できる．limit を計算する際には，バケツのサイズ，トークンの追加速度，および指定されていればピーク速度を考慮する必要がある．latency は，最低限トークンバケツフィルタが処理を行う間隔時間指定していなければ，パケットが処理される前に破棄される．

burst/buffer/maxburst バイト単位のバケツのサイズである．これはある瞬間に利用できるトークンの最大値 (バイト数) である．絞りたい通信速度が大きい場合には，大きなバッファが必要である．バッファが小さすぎると，時間単位あたりに処理しなければならないパケットをバケツに溜めて置くことができない．そのため，処理しなければならないパケットまで破棄されてしまう．トークンバケツフィルタの設定の中で最も重要なパラメータである．例えば，Intel で 10mbit/s を使う場合，この設定速度にするには少なくとも 10kbyte のバッファが必要である．

mpu サイズが 0 のパケットも，使うバンド幅は 0 ではない．イーサネットでは，64 バイト以下のパケットは存在しない．最小パケット単位 (Minimum Packet Unit) は，ひとつのパケットが用いるトークンの最小値を定める．mpu を 1 とした場合，トークンが扱えるパケットは存在しないためすべてのパケットは破棄される．

rate このパラメータによって通信速度の制限を行う。バケツにトークンが入っていて，これを空にすることが許されるとき，デフォルトでは非常に速いの速度でデータがバケツから処理される．rate に設定する値で，他のパラメータのおおよその基準値が決まる．

3.2.4 Modelnet

ModelNet [15] 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。

3.2.5 XENebula

XENebula(ゼネビュラ) とは，三輪ら [16] が提案している模倣インターネットを構築するためのツールセットである．XENebula は，Xen+Nebula の造語である．一般に，多数の PC を連ねたものを PC クラスタ (Cluster) 環境と呼ぶが、Nebula はこれに対し，PC などの構成要素の数や性能についても柔軟に変更可能な環境を指す言葉として，三輪らが提唱しているものである．XENebula は仮想化技術である Xen を用いて Nebula 環境を構築するツールセットである．

XENebula は AnyBED で作った設定をもとに模倣 Autonomous System(AS) 間ネットワークを容易に構築する．インターネット上における AS の繋がりに注目し，AS トポロジとして抽象化したネットワークを構築する．XENebula で構築される模倣 AS 間ネットワークは，一般的な BGP 網として構成される．

3.3 関連研究における問題点

3.3.1 実証環境における問題点

本項では，実証環境を用いて検証を行う上で問題となる事象を 2 つ挙げる．まず実証環境における問題点としてインターネットの特性の有無，実証環境が持つリソース使用の効率について，問題点をそれぞれ挙げる．そして，インターネットの特性を模倣する際に起こる問題点として設定における問題点，限界値における問題点を挙げる．

インターネットの特性の有無

実証環境は，それぞれ特有の機能に特化したものや，汎用的に利用できるように設計されたものなど様々である．しかし，インターネットの特性を考慮すると，インターネットから隔離した実証環境とインターネットを利用した実証環境の 2 つ分類することができる．

インターネットから隔離した実証環境では，インターネットの特性を含まない状態で実験を行う．このような環境では，非常に離れたノード間の通信や，不安定な通信を再現することができない．遅延や帯域の模倣を行っていない場合，音声アプリケーションや，動画のストリーミングのようなリアルタイム通信を行うアプリケーションの十分な検証を

行うことはできない．一方，インターネットを利用した実証環境では，インターネットの特性を含んだ状態で実験を行う．そのため，インターネットを利用している他のユーザやサービスの影響を受けて実験を行うことが可能である．しかし，検証を行うアプリケーションが他のユーザやサービスに与える影響を予め考慮する必要があるため，検証段階としては後期に利用すべきである．また，インターネットを利用した実証環境では，インターネットの特性は一定ではないことから，同じ条件での繰り返し実験を行うことは不可能である．

インターネットの特性を模倣するための方法として，いくつかの方法が提案されている．これらの方法は大きく分けて2つの方法に分けることができる．1つは，専用のハードウェアによる回線網の模倣である．ハードウェアによる模倣は，処理性能は高いが，一装置あたりの価格が高い．大規模な実験ネットワーク上のインターネットの特性を，ハードウェアの模倣のみで構成するのは現実的ではない．

もう1つは，ソフトウェアによるネットワークの模倣である．ソフトウェアによる模倣は，ハードウェアによる模倣と比べ，計測の精度は劣る．そのため，遅延時間や帯域制限の粒度は荒くなる．しかし，ソフトウェア自体は無料で提供されているものが多い．また，ソフトウェアを動作させるノードの処理能力により，限界値は変動する．

これらの方法によって，インターネットから隔離された環境でも，インターネットの特性を創りだすことが可能であるが，実験者が手作業でこれらの構築をしなければならない．これでは，検証実験を行うために多くの時間的・人的コストを支払わねばならず，また手作業によるミスが発生する可能性もある．

リソースの利用率における問題

インターネットには数億にものぼるノードが存在し，各ノードはネットワーク機器に接続されている．その機器間には，遅延や帯域，他のユーザによる背景トラフィックなどが存在する．このような環境を完全に再現するためには，実験環境に多数のノードが用意されている必要がある．そのための環境として実証環境が存在するが，大規模かつ複雑化した実験環境を構築するには問題が残る．実証環境が検証を行うために多数のノードやネットワーク機器を実験設備として提供しているとはいえ，ハードウェア的な上限は必ず存在する．小規模なネットワークを構築した実験であれば，これは問題とならない．しかし，大規模なネットワークを構築する実験となった場合に，インターネットの特性を再現するためには，各ノード間の遅延，帯域などを模倣する必要があるため，膨大なノード数が必要となる．実証環境では多数のノードを実験設備として提供しているが，これらのリソースはノード単位で利用されることが多い．そのため，実験時に利用可能なノードが持つ性能を完全に引き出して実験を行うことは少ない．これはリソースの視点から見たときに，実験環境の規模耐性に問題があると言える．

3.3.2　ネットワーク特性エミュレータにおける問題点

ネットワーク模倣の設定における問題

ネットワーク特性エミュレータを用いることによって，インターネットから隔離された環境においても，遅延，帯域などの特性を生成することは可能である．しかし，ネットワーク特性エミュレータを実験環境に導入するためには各ネットワーク特性エミュレータを設定するための専門的な知識が必要である．実験者は，必ずしもネットワーク特性エミュレータの専門的な知識を持っているわけではない．そのため，実験者が検証を行うためだけにネットワーク特性エミュレータの知識を習得する必要がある．このようなネットワーク特性エミュレータを扱うための技術は，実験の本質ではなく，検証を行うための敷居を上げる要因となる．また，技術の習得のために人的・時間的コストがかかるため，結果的に実験期間を長くとる必要がある．構築するネットワークの規模が大きくなるにつれネットワークの構築に必要なノードの数が増加する．これでは，ハードウェアの限界により，構築する模倣ネットワークの規模に制限が発生する．

リソースの限界における問題

ネットワーク特性エミュレータを用いることによって，遅延や帯域などの特性を模倣することが可能であるが，ノード間の１本のリンクを１台のノードで模倣するだけでは役不足である．ネットワーク特性エミュレータが１本のリンクを模倣する際に，ノードが持つ性能をすべて使い切る必要はなく，模倣を行うに必要なリソースのみを消費する．そのため，１台のノードを用いて１本のリンクを模倣するだけでは，ノードの性能を使い切れない．

１台のノードで複数のリンクを模倣することによって，前述の規模耐性の問題を解消することができるしかし，ソフトウェアによる模倣では，どこまで模倣可能かの限界値が定義されておらず，設定のみが可能であっても模倣するには十分な性能を発揮できるか定かではない．ネットワーク特性エミュレータでノードが模倣可能な限界まで多重化する手法は存在せず，実験者がノードが模倣可能な限界を考慮しながら設定するのは，さらに専門的な知識が必要である．

第4章　リソースを考慮した模倣ネットワークの設計

4.1　インターネットの特性を模倣した実験ネットワークの構築に必要な特性

検証を行うアプリケーションには，実験ノードだけが利用可能なだけではなく，図 4.1 のように，実験ネットワークの中に多数のネットワーク機器が存在している環境を含めて検証を行いたいものがある．そのため，検証を行うための実験ネットワークを構築するためには，多数の実験を行うノードおよび，実験ノードを接続するためのネットワーク環境からなる実験設備が必要である．

また，インターネットには他のユーザやサービス，ネットワーク機器などからの影響を強く受ける．そのため，このようなインターネットの特性を模倣するためには，遅延や帯域，パケットロスなどの模倣を行う機能が必要である．

実験設備に構築するネットワーク上に模倣を行う機能を追加することによって，インターネットの特性を模倣した実験ネットワークの構築ができる．また，模倣を行う特性を遅延，帯域，遅延揺らぎ，パケットロスを別々に模倣することによって，例えば，遅延のみ模倣した実験ネットワークといったインターネットの特性を抽象化した実験ネットワークの構築が可能な StarBED を本研究では用いる．

4.2　本研究で構築する実験ネットワーク

インターネットの特性を模倣した実験ネットワークを構築するためには，何を模倣するのかを定義する必要がある．

2 章で定義したインターネットの特性から，検証するために必要な特性を洗い出す．背景トラフィックを模倣することで，他のユーザやサービスの動作を再現することができる．これによって，検証するアプリケーションが他のトラフィックから受ける影響を検証することができる．背景トラフィックに関しては，梅木 [1] らの研究によって背景トラフィックの生成に成功しているため，本研究では対象としない．

ノード間の遅延を模倣することによって，ノードの距離感を再現することができる．遅延の生成によって，世界の各拠点間の通信のような，物理的に離れた 2 地点間での通信状態を再現して検証することができる．

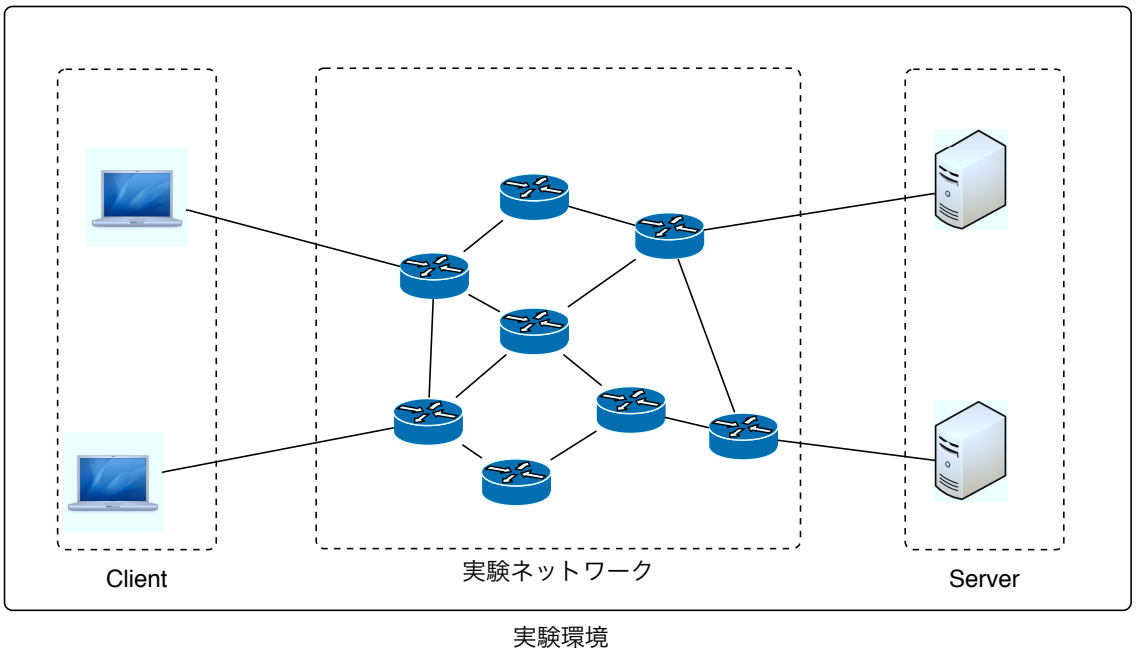

図 4.1: 実験ネットワークを含めた実験環境

インターネット上で利用可能な帯域幅は様々である．これは，背景トラフィックによる圧迫や，中継機器の処理能力の限界などによって制限される．アプリケーションの転送レート制御のため，狭い帯域での通信を行いたい検証のような場合，帯域を制限する手法がよいと考えられる．そして，ノード間の中継機器によるホップ数が存在する．ホップ数は，ノード間に存在するルータの数で変動する．そのため，ノード間の経路が変更した場合，ホップ数も変化する可能性がある．しかし，ノード間のホップ数を考慮したアプリケーションは数少ない．そのため，ホップ数の変化は検証する上で重要ではないと考えられる．一方，IP ルーティングによる経路制御は，実験ノードが所属しているネットワークの分離を行う場合に要求される．ノードが持っているネットワークアドレスが違う場合，実験ネットワーク側でデータの中継を行わなければならない．そのため，IP ルーティングを行う実験ネットワークと行わない実験ネットワークの 2 種類を提供する必要がある．そのため，ノード間の中継機器すべての模倣は行わないが，ルータとして，データの転送を行う機能は必要であり，最低限のホップ数は出現する．これ以降，データの転送機能を IP ルーティングと呼ぶ．これまでのことをまとめると，物理的な距離感の再現のための遅延，転送レート制御の検証のための帯域，不安定な通信環境における検証のためのパケットロス，そして，違うネットワークへの転送機能を提供するための IP ルーティングが本提案で模倣する特性となる．

図 4.2 では，インターネットから隔離された実験ネットワーク上に，遅延，帯域，パケットロスの特性を組み合わせている．これにより，他のユーザやサービスによるトラフィックや，遅延揺らぎのような特性は存在しないが，物理的な距離感や狭い帯域，不安定な通

信を再現した実験ネットワークを構築することができる．このように，実験者が必要とするインターネットの特性を選択することによって様々な特性を模倣した実験ネットワークの構築が可能となる．

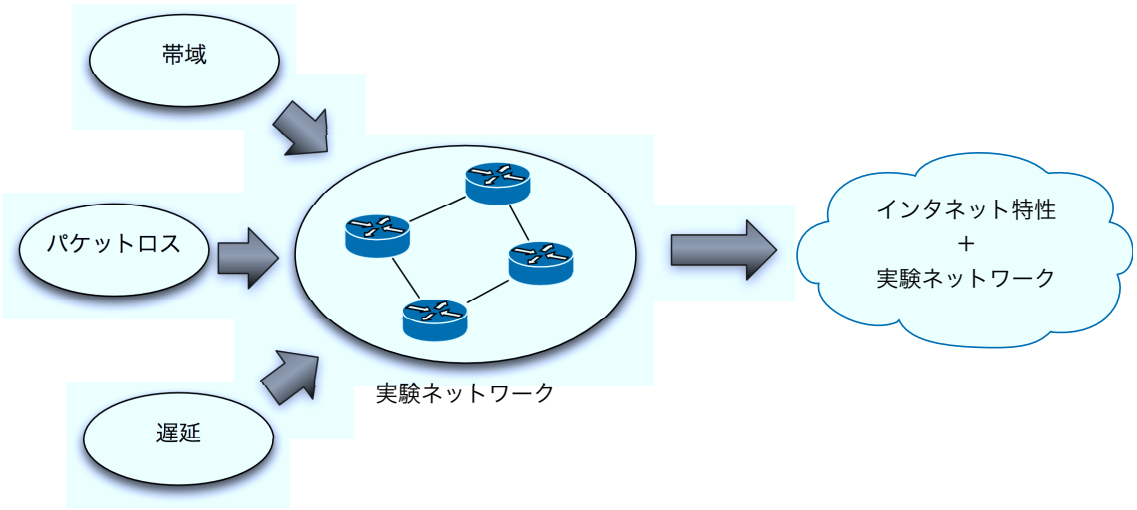

図 4.2: インターネットの特性を模倣した実験ネットワーク

4.3　インターネットの特性の効率的な模倣手法

4.2 節では，インターネットを利用する上で他のユーザやサービス，ネットワーク機器から受ける影響について述べた．これらの遅延，帯域，遅延揺らぎ，パケットロスなどの影響を本研究では，インターネットの特性と呼ぶ．本節では，インターネットの特性を模倣する方法について述べる．

4.3.1　インターネットの特性の模倣

遅延や帯域などのインターネットの特性を模倣する方法は，ハードウェアによる模倣とソフトウェアによる模倣が考えられる．

ハードウェアによる模倣は，遅延網と呼ばれる実際に模倣したい遅延が発生する長さのネットワークを構築するものや，遅延や帯域の模倣を行うために開発された専用機器を用いるものなどがある．ハードウェアを用いて模倣を行う際のメリットは，パケットの送信タイミングや，設定可能な遅延時間が細かいため精度の高い模倣が可能である事が挙げられる．デメリットとして，新たに専用ハードウェアが必要となるため，経済的コストが大きいことが挙げられる．また，物理的なノードの配置に影響を及ぼすため，ネットワーク

構築のための人的・時間的コストも大きくなる．そのため,ネットワークの規模が大きくなるにしたがい実験ネットワーク上への導入が困難になる．

ソフトウェアによる模倣の場合には，NetEm や NISTNet，Dummynet といった Linux，BSD 系 OS で提供されているソフトウェアを用いることによってインターネットの特性を模倣する．ソフトウェアを用いることによるメリットは，実験ネットワーク上で利用できるノードを用いて模倣が可能である事が挙げられる．また，ハードウェアによる模倣手法と比較し，柔軟にリンク特性の変更が可能である．一方デメリットとして，ハードウェアと比較し精度が劣ることが挙げられる．しかし，実験者がネットワーク構築のためのリソースを用意する必要がなく，さまざまなネットワーク環境を構築できるメリットが大きいと考え，本研究では，ソフトウェアによる模倣を用いる．

ソフトウェアによるインターネットの特性の模倣を行う際，実験ネットワークのどこで行うかを考える．インターネットの特性を模倣する場所は実験ノードと実験ネットワークを構成するノードのどちらかで模倣することが考えられる．実験ノードへの負荷が少ないアプリケーションの検証であれば，実験ノードで模倣しても問題はない．しかし，実験ノードへの負荷が大きいアプリケーションの検証を行う場合，インターネットの特性を模倣するソフトウェアと競合する可能性がある．汎用的な実験ネットワークを構築するのであれば，インターネットの特性を模倣するソフトウェアは実験ネットワーク側で模倣を行う方がよいと考えられる．

4.3.2 ネットワーク特性エミュレータの多重化

本研究では，ネットワーク特性エミュレータを用いてインターネットの特性を模倣する．しかし，1 台のノードが大規模な実験ネットワークの特性をすべて模倣することはできない．そのため，前節でインターネットの特性を抽象化した．インターネットの特性を抽象化したことにより，検証に必要のない特性は模倣しない．そのため，実験ネットワーク構築のためのリソースを抑えることができる．しかし，抽象化したインターネットの特性でも 1 台のノードですべて模倣することはできない．そこで，ノードが持つリソースを可能な限り利用するよう多重化する．インターネットの特性の模倣に必要なリソースがノードのリソースを超えた場合，リソースがまだ利用可能なノードを用いて模倣を行う．これによって，ノードのリソースを可能な限り利用して大規模な実験ネットワークを構築することが可能となる．1 つのノード上で複数の模倣を行う方法は 2 つ考えられる．

1. ノード上に仮想 OS を複数起動することによる多重化をしての模倣
   この方法では，ノード上に複数の仮想 OS を構築する．図 4.3 に示すように，仮想 OS はそれぞれソフトウェアによるインターネットの特性の模倣を行う．メリットとして，仮想 OS 間での通信を行うことができる．デメリットとして，仮想 OS を複数起動するため，CPU やメモリなどのリソースを多く消費してしまう．XENebula はこの方法を用いて 1 つの仮想 OS を 1AS として模倣インターネットを構築している．

2. ソフトウェア上で複数のリンクを多重化して模倣
   こちらの方法では，図 4.4 に示すように，実ノード上にソフトウェアを動作させる．ソフトウェアは複数のリンクを同時に処理する．メリットとして，仮想 OS の場合と比較し，仮想 OS を起動するためのリソースをソフトウェアによる処理で利用できる点が挙げられる．デメリットとして，ノード上では 1 つの OS しか動作していないため，ノードへの設定が複雑になることが挙げられる．

本研究では，可能な限り大規模な実験ネットワークの構築することを目的としている．そのため，仮想 OS による多重化では，仮想 OS を起動するためのリソースを消費してしまうデメリットは大きいと考える．したがって，実験ネットワーク構築には，余分なリソース消費を抑えることができるソフトウェアによる多重化を採用する．

図 4.3: 仮想 OS を用いたインターネットの特性模倣の多重化

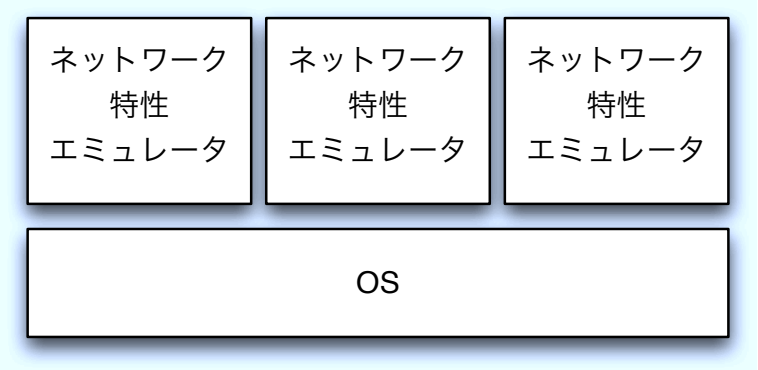

図 4.4: 実ノード上でソフトウェアによる多重化

4.3.3 ノードが持つリソースの定義

ノードが模倣可能なインターネットの特性の限界を考慮するため，インターネットの特性の模倣に必要なリソースの定義を行う．一般的に，ネットワークソフトウェアを利用するうえで消費されるリソースを挙げる．

CPU

CPU は，インターネットの特性を模倣する際に，データ処理のために利用される．ネットワーク特性エミュレータが要求する処理を考慮した場合，ギガヘルツの処理能力を持つ CPU であれば，少なくとも 1000 以上の多重化ができる．このことを考慮すると，CPU リソースは，他のリソースと比較し不足することは考えにくい．そのため，本研究では，CPU リソースを考慮しない．

メモリ

メモリは，インターネットの特性を模倣する際に，データの確保に利用される．メモリが不足すると，データの確保ができず，転送するべきデータを破棄してしまう．そのため，ノードが持つメモリ量は考慮すべきリソースとして重要である．

ネットワークインターフェイス

ネットワークインターフェイスは送受信されるデータのネットワークインターフェイスにはメディア速度と呼ばれる送受信可能な最大処理性能が決められている．一般的には，10Mbps，100Mbps，1000Mbps の 3 種類となる．ネットワークインターフェイスが処理するトラフィックを多重化した場合，トラフィックを流しているノードが持つネットワークインターフェイスも同じ処理能力を持っている可能性が高い．そのため，ネットワークインターフェイスが処理できる能力を超えての多重化を行わないよう考慮する必要がある．

リソースの限界を考慮したインターネットの特性の模倣

前節では，ノードが持つリソースについて述べた．本節では，インターネットの特性の模倣に必要なリソースとノードが持つリソースを考慮したアロケートについて述べる．

本研究でインターネットの特性を模倣するために考慮するリソースは，メモリとネットワークインターフェイスである．実験ネットワークを構築するノードにインターネットの特性を模倣させる際に，必要となるメモリ，ネットワークインターフェイスのリソースを計算する．

図 4.5 で示しているような，Link-3 のインターネットの特性をノードが模倣する場合，ノードが持つメモリ資源はほとんど他のインターネットの特性の模倣に利用されている．

そのため，このノードがこれ以上インターネットの特性を模倣出来ないことを示しているため，他のノードがインターネットの特性の模倣を行うよう設定する．

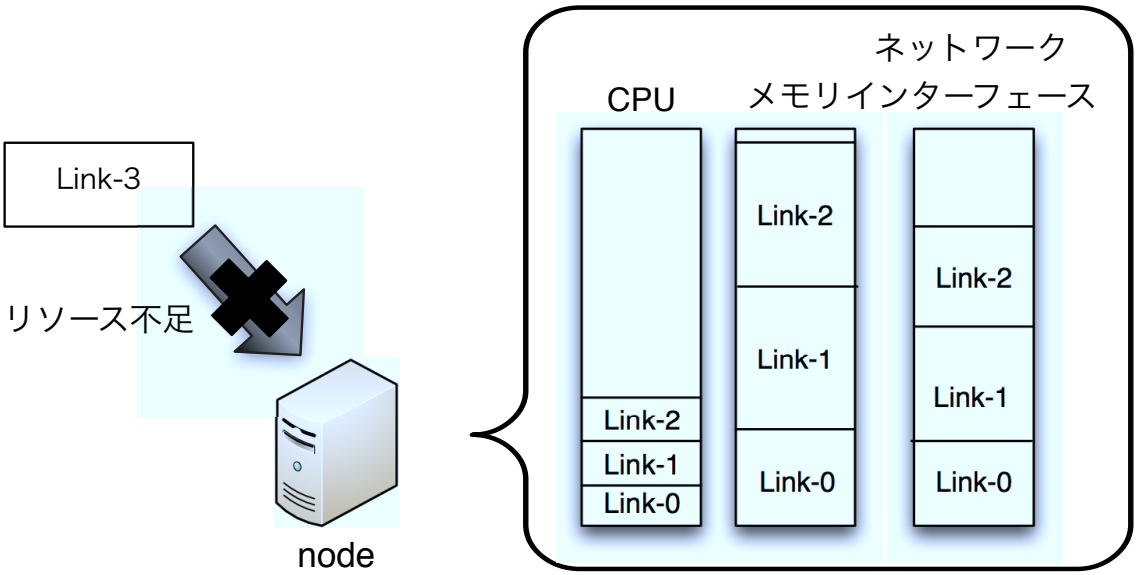

図 4.5: ノードの限界を考慮したインターネットの特性の模倣

4.4 実験ネットワーク構築のための構成

本研究で提案した実験ネットワークを構築するためのシステムの概要図を図 4.6 に示す．

本システムでは，実験ネットワーク構築のために，Control_Node と実験設備，実験設備の情報を把握しているデータベースサーバの 3 つに分類する．Control_Node は，大規模な実験ネットワークの構築を容易に行えるよう，実験ネットワーク構築の制御用ノードとして実験者が操作する．実験設備は，実験を行うノードや実験ネットワークの構築に用いられるノード，ネットワーク機器の集合である．データベースサーバは，実験設備のノードについての情報を収集するためのデータベースとして用いる．データベースサーバが把握する情報は，ネットワークインターフェイスの情報とネットワーク機器への接続情報である．

実験ネットワークの構築手順を図 4.7 に示す．実験者は実験ネットワーク構築の制御を行う Control_Node を操作して実験ネットワークの構築を行う．

4.4.1 ノードの情報収集

Control_Node は，実験ネットワーク構築時に，まず実験設備の情報を収集する．ここで得られる情報には，ノードのネットワークインターフェイス情報とネットワーク機器と

の接続情報を収集する．ネットワークインターフェイス情報とネットワーク機器との接続情報を把握することで，実験設備全体の物理トポロジを把握する．また，実験に用いるインターフェースを把握することで，iLO や KVM のようなノードを管理するためのインターフェースを除外する．実験ノードのメモリリソースは，実験設備のノードと通信を行い，利用可能なメモリ量を収集する．

4.4.2 模倣するインターネットの特性が必要とするリソースの計算

次の手順として，模倣するインターネットの特性に必要なリソースの計算を行う．実験ネットワークに模倣するインターネットの特性に必要なメモリ量の計算は，主に 2 つの特性を模倣する際に必要となる

- 遅延生成
- 帯域制限

遅延生成時に必要となるメモリは，ネットワークインターフェイスに流入される最大のトラフィック量と考慮し，以下の式で定式化できる．

$$\text{メモリ [MByte]} = \frac{\text{トラフィック量 [Mbps]}}{8} \times \text{遅延時間 [sec]} \tag{4.1}$$

これは，遅延生成を行うために，遅延時間分だけトラフィックをノード自身が保持する必要があるため，トラフィック量を Byte に換算したものと遅延時間との積で表すことができる．帯域制限時に必要となるメモリ量は，帯域制限を行うアルゴリズムに従うため，実際に用いる帯域制限方法によって変化する．そのため，ここでは定式化を行わない．パケットロスでは，処理と同時にパケットを破棄するため，メモリリソースは消費されない．そのため，インターネットの特性を模倣するために必要なメモリ量は，遅延生成に必要なメモリ量と帯域制限に必要なメモリ量の和で求めることができる．インターネットの特性を模倣するために必要なメモリ量がノードが持つメモリ量を超えない限りは，メモリリソースは利用できる．

ネットワークインターフェイスが持つリソースは，メディア速度であると本章で述べた．このリソースはネットワークインターフェイスに流入すると考えられる最大のトラフィック量から限界が判断できる．帯域制限を行ったリンクを複数多重化した結果，ネットワークインターフェイスの処理性能を超えてしまった場合，ネットワークインターフェイスはパケットを処理できずに破棄する．

したがって，ノードが持つメモリ量を超えるか，またはネットワークインターフェイスの処理性能を超えるまでが多重化の限界と判断できる．本システムでは，この限界まで模倣の多重化を行う．これらの多重化を行うよう配置された情報が記述されたデータをトポロジーデータとして出力する．

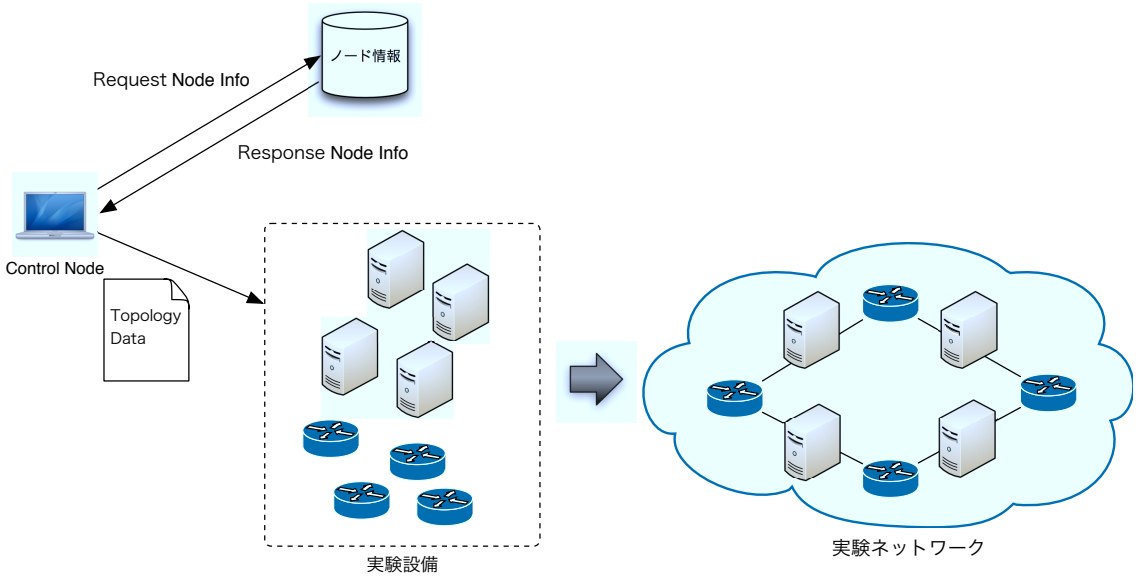

図 4.6: システムの概要図

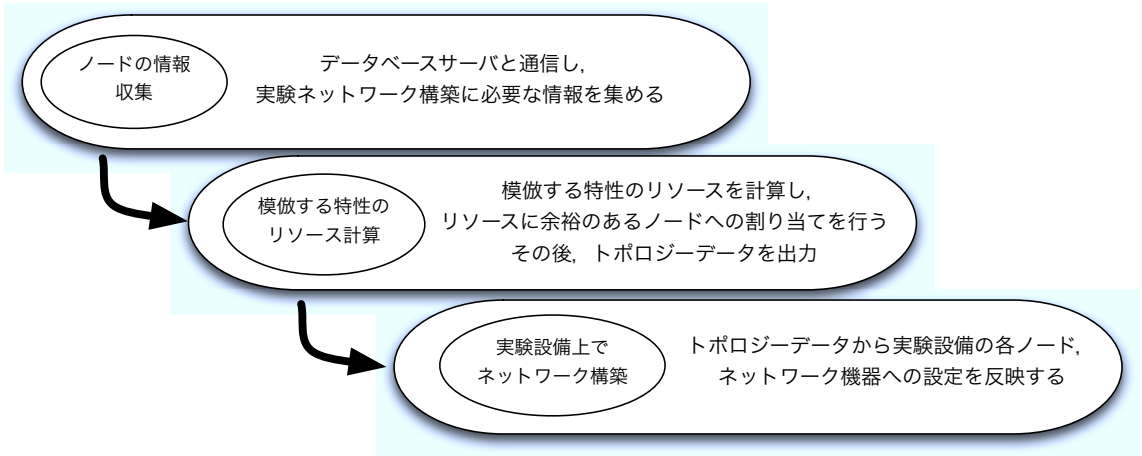

図 4.7: 実験ネットワークの構築手順

4.4.3 ノード，ネットワーク機器への設定

これまでのプロセスによって出力されたトポロジーデータを元に，実験設備のノードとネットワーク機器への設定を反映する．実験者が，実験の度にノード間のケーブルを手作業で接続しなおすのは，ミスを誘発する原因となる．そのため，VLAN を用いた仮想トポロジを構築することで，物理的な配線作業のコストを削減する．また，ネットワーク機器への VLAN の設定を実験者が行うこともミスを誘発する原因となる．そのため，ネットワーク機器への設定は，制御用ノードから一括管理で行う．これにより，人的ミスを防ぐとともに，実験ネットワーク構築の時間的コストを削減することができる．

4.4.4 実験ネットワークの構築形態

4.2 節にて，ノードのリソースを考慮したインターネットの特性の模倣方法を述べた．そして，4.3 節でネットワーク機器への設定方法を述べた．この 2 つのプロセスを行うことで，インターネットの特性を模倣した実験ネットワークを構築することができる．

実際に構築する実験ネットワークの形態として，4.1 節で IP ルーティングを行うものと行わないものが必要であると述べた．この 2 つの形態を実験ネットワークの構築に反映した．

IP ルーティングを行わない実験ネットワークを図 4.8 に示す．Network-0 の中に含まれているノードは同じネットワークに属しており，双方向の通信を行うことができる．Network-0，1 との間はネットワーク特性エミュレータでは IP ルーティングを行われず，Network-0 と Network-1 のノード間の接続性は実験ネットワーク側では提供していない．そのため，1 対 1 の通信を想定した検証や，検証を行うソフトウェアが IP ルーティングを行う場合に利用できる．例えば，ルーティングソフトウェアや，単純な性能試験などがこの実験ネットワークを利用すると考えられる．

IP ルーティングを行う実験ネットワークを図 4.9 に示す．Network-3 に属している実験ノードは Network-0 の実験ノードと通信を行うために，ネットワーク特性エミュレータが動作しているノードである NE-3 をゲートウェイとしてパケットを送信している．NE-3 は Network-3 から Network-0 へのパケットを対応するノード NE-0 へ転送する．Network-0 と接続しているノード NE-0 は NE-3 から転送されてきたパケットを Network-0 のノードへと転送する．この実験ネットワークでは，実験ネットワーク側で IP ルーティングを行うため，実験ノードは相手のノードまでの経路情報を持っている必要はない．そのため，IP ルーティングは行わないが，多数のノードを通信を行う検証の際に利用できる．例えば，アプリケーションマルチキャストや P2P アプリケーションのような分散アプリケーションがこの実験ネットワークを利用すると考えられる．

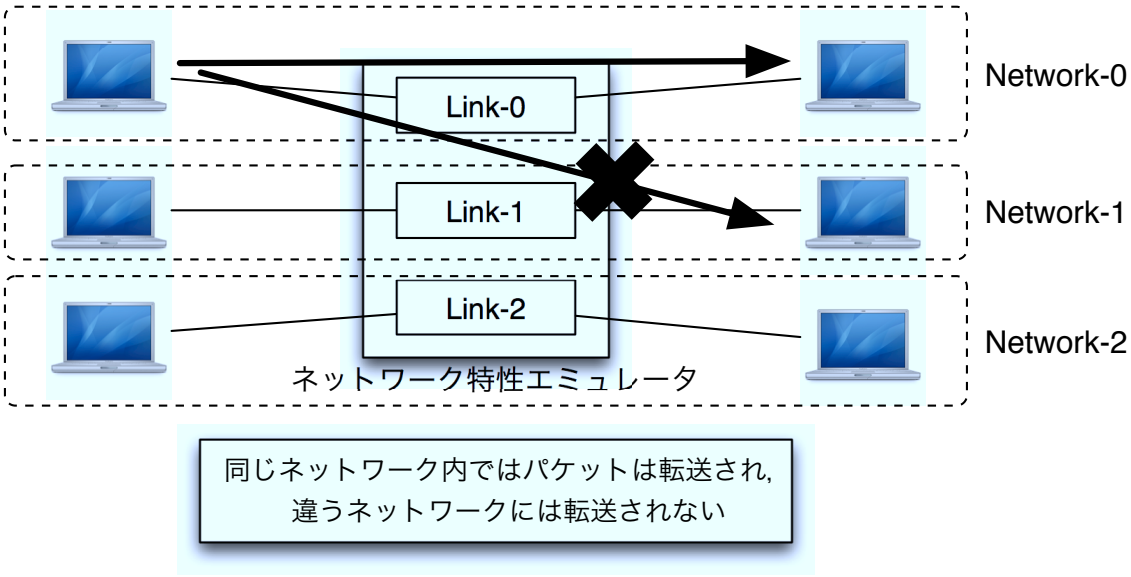

図 4.8: IP ルーティングを行わない実験ネットワーク

4.5 まとめ

本章では，インターネットの特性を考慮した実験ネットワークの構築手法を提案した．提案手法を実現するためのソフトウェアやシステムの構成などについて 5 章では，提案手法を実現するための実装および，利用可能な既存技術について述べる．

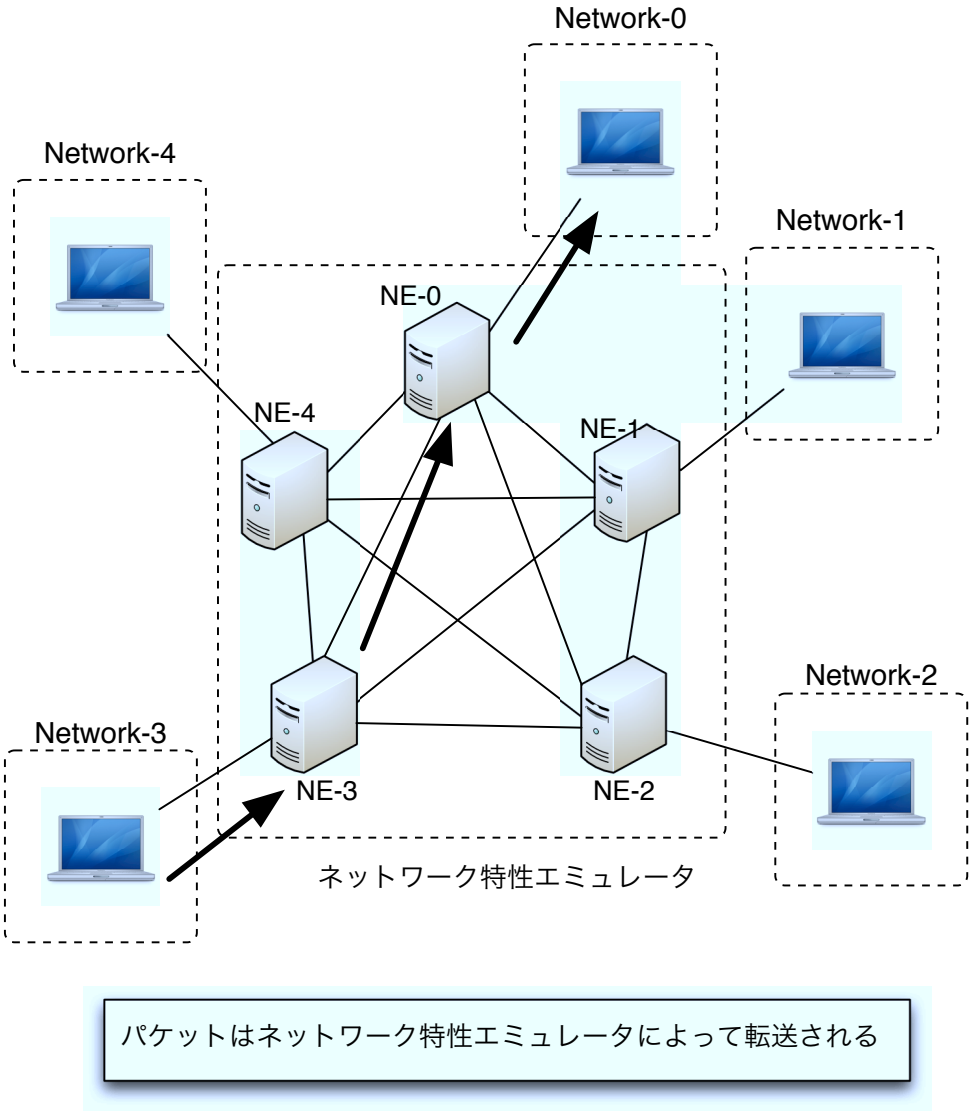

図 4.9: IP ルーティングを行う実験ネットワーク

第5章　実装

5.1　ネットワーク模倣を行うソフトウェア

本研究では，インターネットの特性を持つネットワークを構築するためのソフトウェアとして，実証環境で提供されているノードを利用できるソフトウェアによる模倣を用いる．ソフトウェアによる模倣では Linux や BSD 系 OS で利用できるものが存在するが，本研究では以下の理由により Linux で利用できる NetEm を用いる．

- 最新の OS での利用が可能
  FreeBSD で提供されている dummynet は最新のバージョンでも利用可能であるが，NIST Net は Linux のカーネル 2.6.14 で開発が終了している．

- 遅延揺らぎの設定
  NetEm は遅延を設定した値から上下に揺らぎをつけて模倣することが可能である．dummynet には遅延揺らぎを設定する機能はなく，擬似的に複数の遅延の設定を記述し，確率的に振り分けることしかできない．

- 帯域制限機能の豊富さ
  NetEm 自体は帯域制限の機能を持っておらず，外部機能を用いることによって帯域制限を実現している．そのため，帯域制限を行うための選択肢が複数あり，状況に合わせた帯域制限の手法を選択可能である．

一方，NetEm を使用することによるデメリットを以下に示す．

- ネットワーク模倣の設定を物理デバイスごとにしか設定できない
  NetEm では Qdisc と呼ばれる仮想デバイスを用いることによってネットワーク模倣を実現している．この Qdisc は物理デバイスかもしくは VLAN インターフェースにしか存在しない．そのため，IP alias を用いた 1 つのインターフェースに複数の IP アドレスを用いる用途には向かない．

- デバイスごとに記述可能な設定の上限
  NetEm が用いる Qdisc には Prio と呼ばれるバンドが存在する．デバイスに設定する NetEm のルールはこのバンドごとに設定されるため，デバイスが利用可能なバンドの数に制限される．これを回避するための方法としてバンドを利用しない帯域制限の方法も存在するが，CPU への負荷や，各トラフィックへの公平さに欠けるなどのデメリットが存在する．

5.2 ネットワーク模倣に必要な要素

物理資源として，ノード，スイッチやルータといったネットワーク機器の存在を考慮する必要がある．論理資源として，VLAN ID の存在を考慮する必要がある．StarBED の実験設備は，ノードやネットワーク機器同士がネットワークケーブルで物理的に接続されることによって構成されている．そのため実験者はスイッチの VLAN を用いた論理トポロジを構築し実験を行う．本研究では，VLAN を用いた論理トポロジの中にネットワーク特性エミュレータを導入しノード間の遅延や帯域などを模倣する．

5.3 ネットワーク特性エミュレータのパラメータ

ネットワークの模倣を行うには，実験者が想定しているインターネットの特性をパラメータとして用意する必要がある．これには，ノード間の遅延，遅延揺らぎ，利用可能な帯域幅，パケットロス率が挙げられる．NetEm を用いてエミュレートを行うには，その他にも帯域制限を行うために必要なキューサイズを設定する必要があるが，これは入力される帯域の値より計算可能であるため，実験者が設定する必要はない．最終的に本研究でインターネットの特性の模倣に必要とされるパラメータは，ノード間の遅延，遅延揺らぎ，帯域，パケットロス率の 4 つとなる．

5.4 ネットワーク特性エミュレータの展開手法

ネットワーク特性エミュレータを用いて効率的に模倣を行うための構築手順を以下に示す．

1. ネットワーク特性エミュレータを動作させるノードのリストの作成

2. リスト化されたノードのインターフェース情報の取得

3. 設定ファイルからリンクの模倣に必要なメモリ量の算出

4. 論理トポロジの構築に必要となる VLAN ID の割り当て

5. リンクの模倣に必要なメモリ量が各ノードのリソースを超えないよう配分

6. 5で生成したデータベースに基づき，ネットワーク特性エミュレータを動作させるノードへの設定

7. 5で生成したデータベースに基づき，ネットワーク機器へのVLANの設定

5.4.1 模倣形態

本節では，ネットワーク特性エミュレータをどのように用いて特性を模倣するのかの解説を行う．ネットワーク特性エミュレータでは，データリンク層におけるブリッジ機能を用いた同じセグメント内のリンク特性の模倣と，IPフォワーディング機能を用いた違うネットワークを中継する機能を持つインターネットの特性の模倣を行うことが可能である．

ブリッジ機能を使用する場合、ブリッジ動作を行いたいインターフェースをブリッジインターフェースに収容する必要がある．ブリッジインターフェースは仮想的なインターフェースであり， ブリッジインターフェースに収容したインターフェース間でブリッジ動作が行われる．収容したインターフェースで受信したパケットは，受信時に必要な処理(フィルタ)が行われた後に出力先が決定され，必要であれば出力先の数だけパケットがコピーされる(ブリッジ処理)．出力先が決定したパケットは，ポリシーフィルタで処理されたのち，決定した出力インターフェースから出力される．ブリッジインターフェースは仮想インターフェースであるため，ネットワーク特性エミュレータの設定をそのまま使用することができない．そのため，ブリッジインターフェースに収容したインターフェースにネットワーク特性エミュレータの設定を反映させる．

リンクの模倣

リンク特性の模倣とは，2つのノード間に発生する遅延や，帯域などの模倣を行うことを指す．ネットワークトポロジは図5.1に示すように，2つのノードはIP層では，同じネットワークアドレスを持つ．この形では，1対1の通信を模倣しており，ネットワーク特性エミュレータはデータリンク層でデータのフォワーディングを行う．そのため，Host-AからHost-Bへ通信する際，Host-Aにはデフォルトゲートウェイや，静的な経路情報は必要ない．Host-A，B間はスイッチに接続されているように見える．そのため，Host-A，Bからは中継機器が存在しないように見える．

想定ユースケースとして，リンク特性の模倣を行う実験ネットワークでは，単に1対1の通信のみを行うアプリケーションや，IPルーティングを行うソフトウェアなどがある．単に1対1の通信であれば，IPルーティングを実験ネットワークで行う必要もないため，同じネットワーク内で検証を行えばよい．IPルーティングを行うアプリケーションの場合，実験ネットワーク側でIPルーティングまで提供してしまうと，検証を行うことが出来ないため，リンク特性の模倣を行う実験ネットワークが必要である．

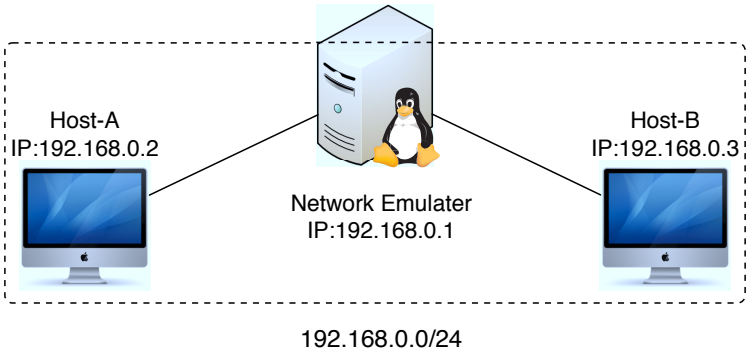

図 5.1: ブリッジ機能を用いたリンク特性の模倣

ネットワークの模倣

この形態では，ネットワーク特性エミュレータに接続されるノードはそれぞれ違うネットワークアドレスを持ち，ネットワーク特性エミュレータを動作させるノードはルータと同じ機能を持つ．ネットワークトポロジは図 5.2 に示すような形となる．Host-A から Host-B へデータを送信する場合，ネットワーク特性エミュレータをデフォルトゲートウェイとして設定するか，静的な経路情報を設定する必要がある．

アプリケーション自身が IP ルーティングを行わず，相手ノードまでの経路を中継ルータに任せることを前提として作成されたアプリケーションの場合，1 対多の通信を行うためには実験ネットワーク側で IP ルーティングまでを提供する必要がある．そのため，アプリケーションマルチキャストや，Skype などに代表される音声，映像アプリケーションで複数のノードと通信を想定したアプリケーションの検証を行うために利用される．

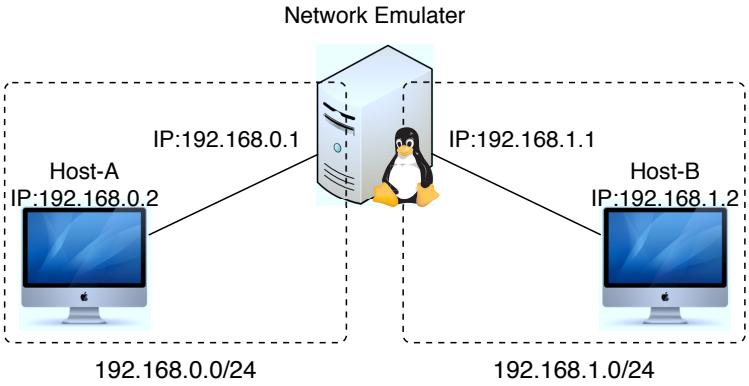

図 5.2: IP フォワーディング機能を用いたリンク特性の模倣

5.5 利用するネットワーク特性エミュレータ

本研究では，ソフトウェアのネットワーク特性エミュレータを 1 つ取り上げ，模倣ネットワークの構築に利用する．dummynet では遅延揺らぎの設定ができないことから Linux で提供されている NetEm を採用した．

NetEm は，帯域制限の機能を持っていないため，ルール記述の方法が違う．遅延生成，遅延揺らぎ，パケットロス，帯域制限の設定を図 5.3 に示す．NetEm のルール記述は木構造のような構造を持つ．NetEm のルール構造を図 5.4 に示す．図 5.4 では，root から派生した先に遅延を 10ms 生成する設定と，遅延 10ms とその 10ms の遅延が 5ms の間で揺れるように模倣する設定が書かれている．handle とはルールの ID 番号であり，どのルールから派生しているのかやそのルールの判別に用いている．ルール ID20 の下には，帯域制限の設定が書かれている．帯域制限は 10Mbps の設定と 50Mbps の設定が書かれている．この段階で，3 つのルールが記述されているが，この NetEm ノードに入ってくるトラフィックをそれぞれのルールに振り分けるルールが帯域制限の下に書かれている．トラフィックの振り分けは filter と呼ばれるコマンドで制御する．図で示している設定では，150.65.117.0/24 からのトラフィックが帯域制限 10Mbps に振り分けわれ，150.65.118.0/24 からのトラフィックが帯域制限 50Mbps のルールに振り分けられる．その他のネットワークから入ってきたトラフィックに関しては，ルール ID10 に振り分けられる．複数のルールを設定する場合には，filter のルールを書く必要がある．

```
# tc qdisc add dev eth1 root handle 1: netem delay 10ms 5ms loss 5%
# tc qdisc add dev eth1 parent 1: handle 10: tbf rate 10mbit latency 1s buffer 10kb
```

図 5.3: 遅延生成，遅延揺らぎ，パケットロスの設定

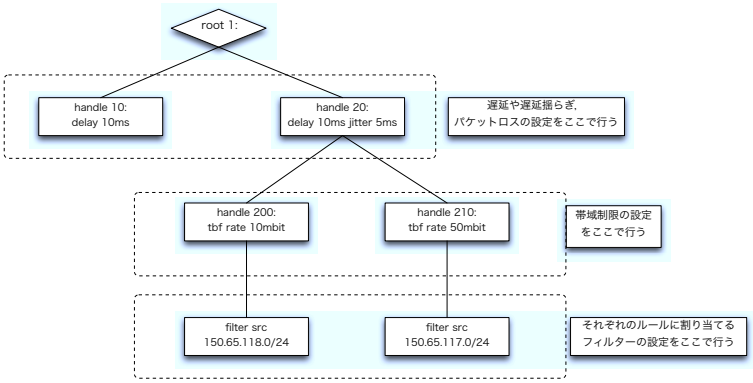

図 5.4: NetEm のルール構造

5.5.1 帯域制限の方法

NetEm は帯域制限の機能を持っておらず，外部機能を取り込むことによって帯域制限を実現しているため，帯域制限を行うための方法が複数提案されている．本研究では，以下の理由により複数提案されている中のトークンバケツフィルタを用いた帯域制限の方法を採用した．

- 単純に帯域の制限を行うのに向いている．
  トークンバケツフィルタは単純な Qdisc で，実験者が設定した速度を越えない範囲で到着パケットを通す処理を行う．トークンバケツフィルタの設定が突発的なバーストを許容しない状態であれば，完全な上限を決定することになり，設定値を超えたトラフィックは破棄される．これは単純なネットワーク機器の挙動に即しており，また観測される帯域の模倣に向いていると言える．

- CPU への負荷が軽く，正確な処理が可能
  トークンバケツフィルタで行われる処理は，バケツに到着するパケットを拾い，送信を行う．トークンバケツフィルタにトークンが存在する場合には，パケットが転送され，トークンが存在しない場合には，パケットは転送されず，バーストを許容しない設定であれば，トークンが存在しない時点でパケットは破棄される．他の帯域制限方法と違い，トラフィックの種類や，確率的なキューイングを行わないため，CPU への負荷は少ない．また，一定の周期でトークンがバケツに到着するため，過負荷状態においても正確な処理が可能である．

- 複雑なパラメータを必要としない
  トークンバケツフィルタを設定するためのパラメータは単純に帯域制限値，バイト単位のバケツのサイズ，パケットがトークンバケツフィルタに留まれる時間の最大値の 3 つでよい．高度なフィルタを用いている他の帯域制限方法では，入力されるパケットの最大，最小，平均サイズやハッシュの計算などが必要となる．これらを決定するためには，NetEm にパケットを入力するアプリケーションがどのようなパケットを送信するのかを知って置く必要がある．また送信されるパケットのサイズが大きく変わるような場合，模倣の正確さは低下する可能性がある．

5.6 システム構成

本システムでは，StarBED のノードをコントロールノードと NetEm ノードに分ける．

コントロールノードは，Netem ノード及びネットワーク機器に設定を反映させるノードである．NetEm ノードはコントロールノードより受け取ったコンフィグを反映させ，特性を模倣した実験ネットワークを構築するノードである．

本稿で提案するシステムは 3 段階

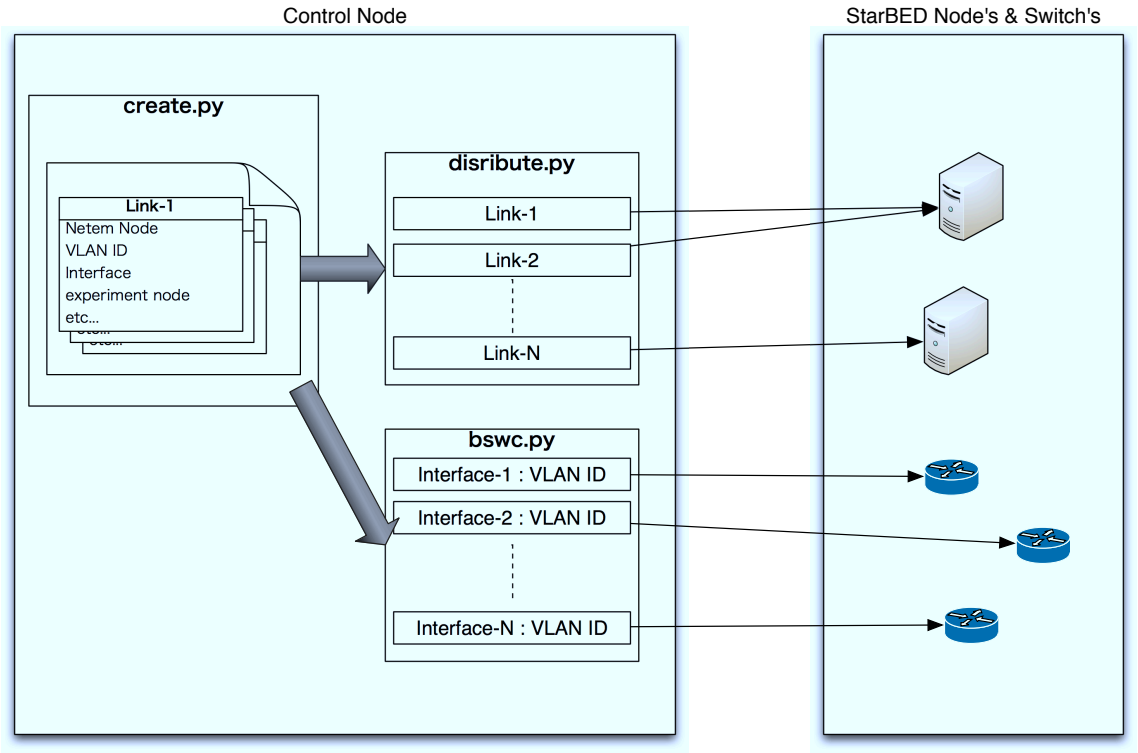
Control Node
StarBED Node's & Switch's
create.py
Link-1
Netem Node
VLAN ID
Interface
experiment node
etc...
disribute.py
Link-1
Link-2
Link-N
bswc.py
Interface-1 : VLAN ID
Interface-2 : VLAN ID
Interface-N : VLAN ID

図 5.5: システムの構成

NetEm を動作させるノード (以下 NetEm ノードと呼ぶ) の選択，及び各リンクを模倣するために必要なリソースの計算，各ノードに展開されるコンフィグの生成および割り当て (図 5.6)，各ノードおよびネットワーク機器への設定反映 (図 5.7) により実験トポロジを構築により構成される．

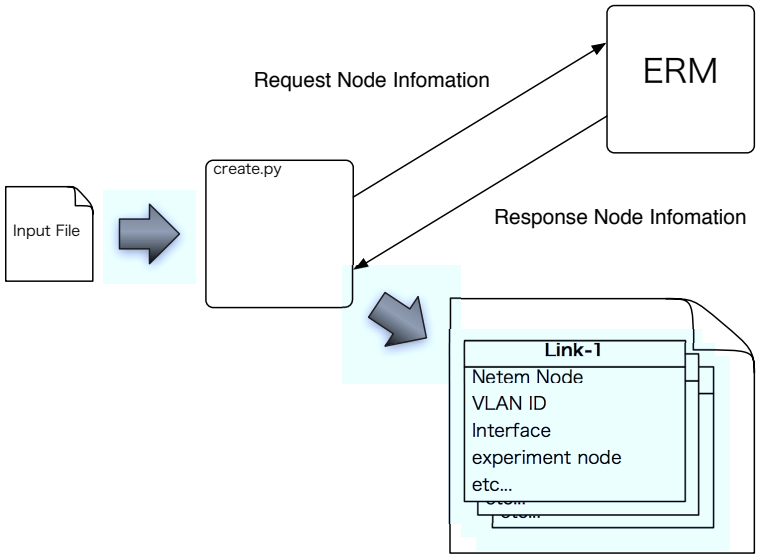

図 5.6: ERM と通信によるインターフェース情報の取得

5.6.1 ネットワーク特性エミュレータノードの選択

実証環境では，利用可能なノードはその時々によって違うため，実験者が利用できる範囲で選ばなければならない．また，実証環境で利用可能なノードすべてをネットワーク特性エミュレータに利用してしまうと，実験に使用するノードがなくなってしまうため，実験に使用するノードを除外する必要がある．

実験に使用せず，ネットワーク特性エミュレータを動作させても問題ないノードの選択が終了した後，実験用インターフェースの選択を行う．StarBED では，各ノードが持つネットワークインターフェイスを管理用と実験用の 2 種類に分けている．これは，一括処理を行うノードからアクセスするためのインターフェースと実験を行うインターフェースを分けることによって，実験トラフィックと管理用のトラフィックが混合することを避けるためである．

このことから，管理用インターフェースを用いて実験を行うことは StarBED では推奨されていないため，ネットワーク特性エミュレータが模倣に使用するインターフェースから除外しなければならない．実験用インターフェースがすべて同一の命名規則に従っているのであれば，実験用インターフェース名は固定で問題ないが，OS の種類によってはインターフェースの命名規則が同じではなく，またノードの修理によってインターフェー

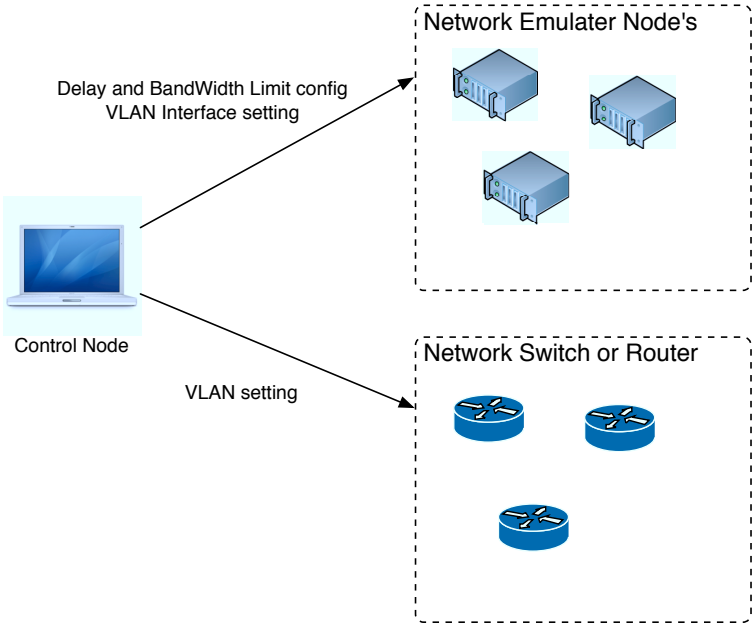

図 5.7: ノードやスイッチへの設定反映

ス名が変化することがある (例 Linux：eth0, eth1 FreeBSD：em0, bge0 など)．そのため，ネットワーク特性エミュレータを動作させるノードを選択した後，実験用インターフェース名を取得する必要がある．

実験用インターフェース名の取得は，SpringOS に含まれている ERM とノードからそれぞれインターフェース情報を取得する必要がある．ERM には，管理用・実験用インターフェースの MAC アドレスが記述されており，インターフェース名は記述されていない．これは前述の OS による命名規則の変化からインターフェース名の記述が除外されている．そのため ERM から実験用インターフェースの MAC アドレスを取得し，対応するノードからインターフェース名とその MAC アドレスを取得することによって，実験用インターフェース名を取得する (図 5.9)．また，この時に実験用インターフェースが接続されているスイッチのポートも取得しておく．これは，後述のスイッチの設定の際に必要となるため，ここで取得している．

図 5.8 に ERM から取得できる情報を示す．name はノードにつけられている名前である．この他利用しているユーザ名である owner や HDD の種類である diskhint などが記述されている．ERM から取得できる情報の中で本研究にて必要なのは if-x と書かれている部分である．ここにはノードが持っているネットワークインターフェースの情報が記述されている．「type=manage」と書かれているインターフェースは管理用インターフェースであることを意味し，「media=GigabitEthernet」はインターフェースがギガビットイーサネットの NIC であることを示している．「MAC=’00:23:7d:d5:67:90’」はその NIC の MAC アドレス，「phy-port=’mgsw4:10/8’」は NIC と接続しているスイッチの名前及びポート

を示している.「IP-addr=172.16.6.200」は管理用インターフェースであればノードの名前と対応付けられている IP アドレスが記述されている．実験用インターフェースである場合には 0.0.0.0 となる.「devname=」は OS 毎にインターフェース名が違うため空欄である.

```
name: h200
owner: JaistInetem-09
state: LEASED
use: 1
Muse: 1
Mowner: JaistInetem-09
Mstate: LEASED
diskhint: SATA
bootdisk: SATA
Power: SNMP-NECMIB
n_if:  3
n_active_data_if:  2
if-0: type=manage media=GigabitEthernet MAC='00:23:7d:d5:67:90' phy-port='mgsw4:10/8' IP-addr=172.16.6.200 devname=
if-1: type=experiment media=GigabitEthernet MAC='00:23:7d:fb:0b:ca' phy-port='exsw8:5/8' IP-addr=0.0.0.0 devname=
if-2: type=experiment media=GigabitEthernet MAC='00:23:7d:fb:0b:cb' phy-port='exsw8:10/8' IP-addr=0.0.0.0 devname=
n_mc:  0
.
```

図 5.8: ERM のノード情報

5.6.2 リソースの計算

ネットワーク特性エミュレータが模倣する際に必要となる資源の算出を行う．遅延生成，帯域制限に必要なメモリ量とノードが持つ 1 つのネットワークインターフェースが物理的に使用できるリンクスピードを考慮する必要がある.

ネットワーク模倣を行うツールとして，NetEm を使用する．NetEm を利用する際に，Linux のカーネルオプションとして用意されている Time Frequency を 1000Hz に設定していることを想定している.

5.7 ノードの要件

本システムでは，各ノードに対して SSH [17] を用いてインターフェースの情報や NetEm の設定を行う．そのため，実験に使用する全てのノードが SSH サーバの機能を有している必要がある．また，NetEm ノードは遅延生成，帯域制限を行うため，Linux のカーネルオプションで，QoS 機能を有効にしておくことと，iproute2 に含まれている tc コマンドをインストールしておく必要がある.

コントロールノードにおいては，本システムが Python で記述されており，JSON 形式のデータを扱うため，Python2.6 が必要となる.

コントロールノードは NetEm ノードに SSH で通信を行うため Python の SSH ライブラリである Paramiko [18] をインストールする必要がある．表 5.1 にコントロールノード，

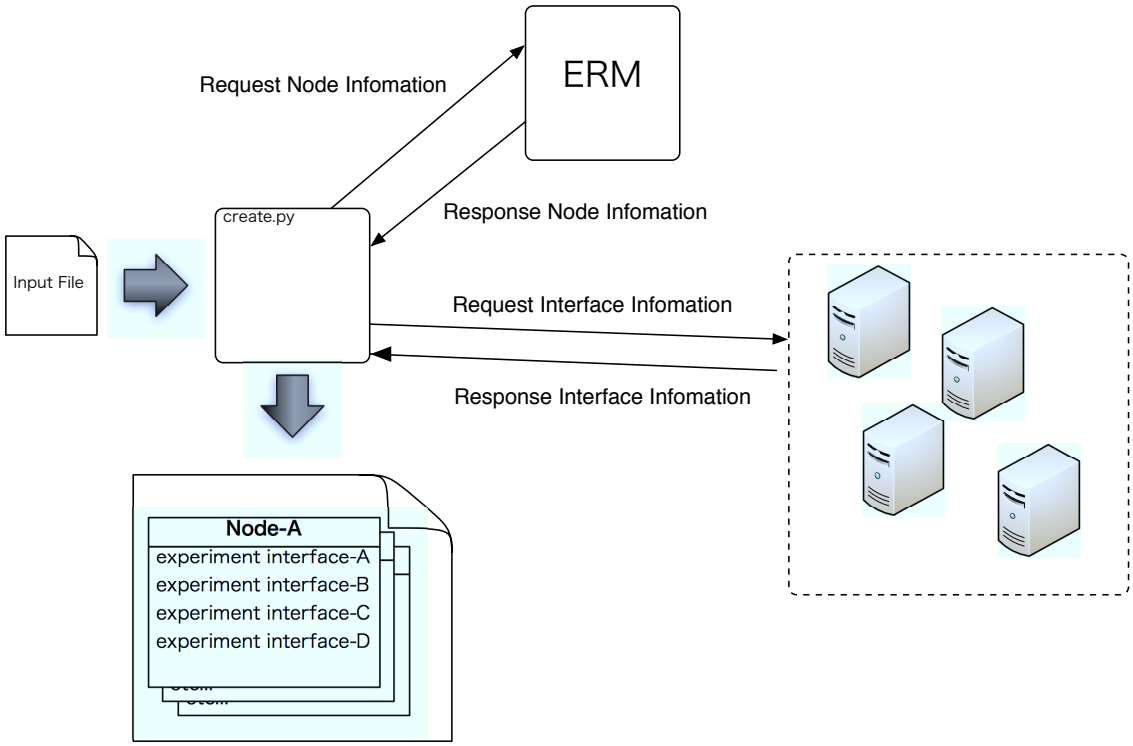

図 5.9: 実験用インターフェース名の取得

NetEm ノード，実験ノードそれぞれに必要となるパッケージやカーネルコンフィグレーションをまとめた表を示す．

表 5.1: 各ノードの要件

ノードの種類	コントロールノード	NetEm ノード	実験ノード
パッケージ	Python2.6 Paramiko	iproute2 sshd	sshd
カーネルオプション	なし	QoS 機能	なし

5.8 ネットワーク構築の自動化

本システムでは，実験者は NetEm ノードを操作する必要はない．コントロールノードに入力したデータより，コントロールノードが模倣するために必要な NetEm ノードの数を計算し，それぞれに設定を反映させる．また，NetEm ノードが接続しているネットワーク機器にも設定を反映させる．

5.8.1 NetEm ノードに割り当てる設定

まず，NetEm を動作させるノード，StarBED で利用可能な VLAN ID のリストを生成する．NetEm ノードのリストは，実験者が明示的に指定した範囲のノードから生成する．この時，NetEm ノードの中から検証実験に使用するノードを除外する．除外した結果，検証実験に含まれないノードを NetEm ノードのリストとして用いる．VLAN ID のリストは，実験者が明示的に指定した範囲の VLAN ID から生成する．

次に NetEm ノードがどれだけのリンクを模倣できるかの計算を行う．NetEm が遅延を生成する方法は，Qdisc デバイスに設定された遅延時間だけパケットを保持し，設定された時間が経過した後，デバイスから送信することによって実現している．そのため，必要となるリンクのリソースは，メモリおよび回線帯域となる．遅延生成を行う際に，NetEm ノードに対してどれだけのトラフィック量が流れてくるかは，実験内容によるため決定的でない．そのため，NetEm に入力されるトラフィック量の最大値は，NetEm ノードが持つ NIC が利用可能な最大帯域となる．本稿では，NIC が利用可能な最大帯域を Link_Speed と呼ぶ．1 本のリンクの遅延生成を行うために必要なメモリを Delay_Memory とした場合，遅延 Delay[sec] のリンクを模倣するために必要なメモリは，以下の式により求められる．

$$\mathrm{Delay\_Memory[MByte]} = \frac{\mathrm{Link\_Speed[Mbps]}}{8} \times \mathrm{Delay[sec]} \tag{5.1}$$

帯域制限の場合は，NetEm がトークンバケツフィルタに処理を任せるため，設定するトークンバケツのサイズによって必要なメモリ量が変化する．トークンバケツのサイズが帯域制限を行うために最低限必要なサイズよりも小さい場合は，指定した値よりも狭い帯域しか利用することができない．逆に，必要以上に大きなサイズに設定した場合は，常に余分なパケットがトークンバケツに貯まるため，想定しない遅延が発生することになる．したがって，指定した通りの帯域制限を行い，かつ遅延の発生を最小限に抑えるようトークンバケツのサイズを適切な値に設定する必要がある．

帯域制限を行うために必要なメモリを Bandwidth_Memory とした場合に，帯域 B[Mbps] のリンクを模倣するために必要なメモリは以下の式より求められる．

$$\mathrm{Bandwidth\_Memory[MByte]} = \frac{\frac{B[\mathrm{Mbps}]}{8}}{1000[\mathrm{Hz}]} \tag{5.2}$$

あるノード間1の遅延，帯域を模倣するために必要なメモリ量は，それぞれの遅延，帯域を模倣するために必要なメモリ量である．そのため，実験ネットワークに存在するすべてのノード間の遅延，帯域を模倣するために必要なメモリ量を NetEm_Memory とした場合，以下の式によって求めることができる．

$$\mathrm{NetEm\_memory[MByte]} = \sum_{l=0}^{k} \mathrm{l'sDelay\_Memory[Mbyte]} + \mathrm{l'sBandwidth\_Memory[MByte]} \tag{5.3}$$

NetEm_Memory が NetEm ノードで利用可能なメモリを超えた場合には，そのノードでリンクを模倣することができないことを示しているため，次のノードの選択を行う．また，NIC の Link_Speed をリンクが利用する合計帯域を超えた場合にも，NIC の物理的限界を超えているため，次のノードの選択を行う．

各リンクには StarBED から提示された VLAN ID を設定する．これにより，異なるリンクのトラフィックが混同することを防いでいる．

このプロセスでは，最終的に 1 本のリンクに，NetEm ノードの IP アドレス，ブリッジインターフェース名，2 つの VLAN インターフェース名，リンクに適用する NetEm のコマンド，リンクの両端に接続するノードが記述される．

これらの処理を終わらせた後，すべてのリンク情報を集約したリンクデータを出力する．

5.8.2 Netem の設定ファイル

NetEm ノードのメモリ計算が終了した後，要求されているリンクを模倣するための NetEm の設定コマンドを生成する．遅延生成，遅延ゆらぎ，パケットロスに関しては，NetEm そのもので模倣可能であるため 1 つの設定で生成可能である．帯域制限に関しては，NetEm 自体が帯域制限の機能を持っていないため，帯域制限を行うモジュールにデー

タを渡す設定を記述する必要がある．そのため，NetEm で遅延生成，帯域制限の両方を設定したい場合は 2 つの設定を記述する必要がある．

図 5.10 に生成される設定ファイルを示す．設定ファイルは JSON(JavaScript Object Notation) 形式で記述されており，Link-No，実験ノード，NetEm の設定，NetEm ノードの IP アドレス，NetEm ノードが生成するブリッジインターフェース名，ブリッジインターフェースに所属するインターフェース名が記述されている．Link-No はリンク特性の模倣を行うリンクの識別子として用いている．実験ノードはリンクに接続される実験ノード名を記述している．StarBED で提供されているノードが接続される場合には，DNS に登録されているノード名が記述され，外部機器を用いている場合には，None と記述される．NetEm の設定は，NetEm ノードが模倣を行うリンクの設定コマンドが記述されている．netem と書かれている箇所は，遅延，遅延揺らぎ，パケットロスの設定が記述されており，tbf と書かれている箇所にはトークンバケツフィルタの設定が記述されている．netemip には NetEm ノードとして割り当てられたノードの IP アドレスが記述されており，この IP アドレスを持つノードがリンク特性の模倣を行う．bridge-if は Link-No の番号が割り当てられたブリッジインターフェース名である．リンク特性の模倣では，接続される実験ノードが同じネットワークに所属する必要がある．そのため，ブリッジインターフェースを作成し，データリンク層で実験ノード間のデータをフォワーディングできるように設定する．physical-if は作成した bridge-if に所属する物理インターフェースを記述している．これは基本的に，VLAN インターフェースとなる．本システムでは，NetEm の多重化を行うため 1 つのインターフェースに複数のトラフィックが流れてくる．そのため，VLAN インターフェースを作成し，それぞれのトラフィックが混合しないようにしている．

```
{
   "Link0": {
      "node": ["a159", "a160"],
      "netem": {
         "netem": "netem limit 10000 delay 80ms 10ms loss 10%",
         "tbf": "tbf rate 42mbit latency 1s buffer 42kb"
      },
      "netemip": "172.16.6.200",
      "bridge-if": "br0",
      "physical-if": ["eth0.760", "eth0.761"]
   }
}
```

図 5.10: NetEm の設定ファイル

```
{
   "150.65.117.41": {
      "route": {
         "route": {
         },
         "netemip": "192.168.254.1",
         "netemif": "eth1"
      },
      "eth0.201": {
         "1": {
            "10": "netem limit 10000 delay 100ms",
            "child": {
               "100": "tbf rate 10mbit latency 1s burst 10k",
               "child": {
                  "flow": "parent 1: protocol ip prio 1 u32 match ip src 192.168.0.0/24"
               }
            }
         },
         "dst": "192.168.1.0/24"
      },
      "eth0.200": {
         "1": {
            "10": "netem limit 10000 delay 100ms",
            "child": {
               "100": "tbf rate 10mbit latency 1s burst 10k",
               "child": {
                  "flow": "parent 1: protocol ip prio 1 u32 match ip src 192.168.1.0/24"
               }
            }
         },
         "dst": "192.168.0.0/24"
      }
   }
}
```


図 5.11: test

5.8.3 NetEm の反映

各 NetEm ノードに設定を反映させるプロセスの解説を行う．コントロールノードは Python のパッケージとして提供されている Paramiko を用いて，SSH の通信を行う．コントロールノードと NetEm ノードとの間に SSH のセッションを確率した後，NetEm ノードに反映させるべきコマンドを送信する．SSH のセッションを確立する際には，ノードのログインするためのパスワードまたは SSH 鍵を指定する必要がある．また Root ユーザでログインを試みるため，sshd_config の Permit Root Login を Yes に指定する必要がある．

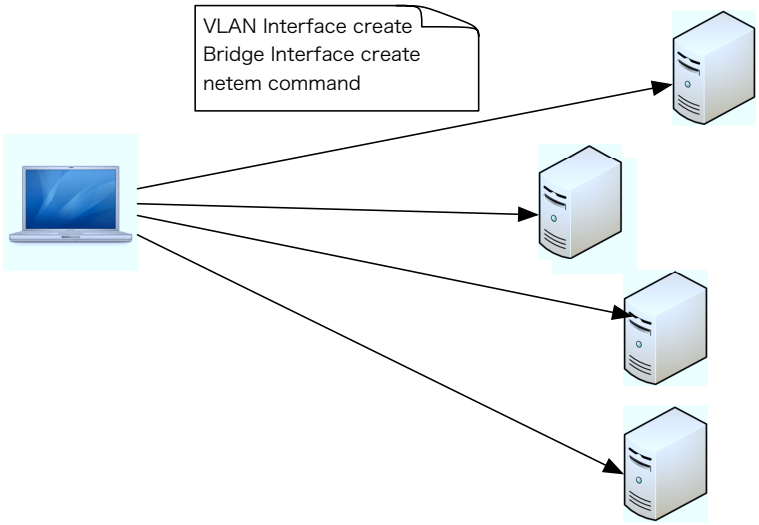

図 5.12: インターフェース，NetEm の設定の反映

5.8.4 VLAN の設定

NetEm ノードの設定が終了した後，NetEm ノードが接続しているネットワーク機器の設定に移る．ネットワーク機器の設定は NetEm ノードで作成した VLAN インターフェースを接続するための設定を行う．実験ネットワークの VLAN の設定は SpringOS に含まれる SWMG を使用して行う．SWMG は，Cisco や Foundry，D-Link などのスイッチのコマンド体系を吸収し，スイッチに VLAN の設定を反映させるソフトウェアである．

各ノードに設定されたリンクに割り当てられた VLAN ID をスイッチにも反映させる．これによって NetEm ノードに設定されたインターフェースと実験を行うノード間の接続性を確保する．SpringOS1.5 以降の SWMG であればタグ VLAN を設定することが可能であり，スイッチ間リンクのタグ VLAN まで設定可能である．

本システムでは，SWMG のクライアントとして動作し，設定ファイルに記述されている VLAN ID を NetEm ノードが接続しているスイッチのポートに設定する．実験用ノードの実験用インターフェースが接続しているスイッチのポートへの設定は untagged VLAN

の設定を反映させるよう実装しているが，これは実験者が行う実験の内容によって変化するため，オプション指定としている．

第6章　評価

6.1　インターネットの特性を模倣した実験ネットワークの構築時間

検証を行う環境として，インターネットの特性を模倣した実験ネットワークの構築時間は可能な限り短いほうがよい．そこで，インターネットの特性を模倣した実験ネットワークの構築にかかる時間の測定した．測定する実験ネットワークの構築時間は，実験者が遅延や帯域などのインターネットの特性をパラメータとして記述したファイルをシステムに入力した時間から，NetEm ノードとネットワーク機器への各種設定が終了する時間までである．構築する実験ネットワークのトポロジを図 6.1 に示す．NetEm に接続される各ノードは，それぞれ 1 対 1 の接続関係であり，各リンクの特性は，衛星通信が 500ms であることを考慮し，10ms〜500ms までの範囲でランダムに生成した．帯域制限値は，映像配信サービスであるアクトビラなどが約 12Mbps の帯域幅を要求することを考慮し，12Mbps までの 1M きざみでランダムに生成した．IP ルーティングを行わず L2 までのインターネットの特性を模倣した実験ネットワークの構築時間を表 6.1 に示す．

表 6.1: 模倣ネットワークの構築にかかる時間 [sec]

リンク数	試行回数 1	試行回数 2	試行回数 3
5	13.774	13.848	13.883
10	16.834	16.857	16.963
15	20.133	20.108	20.142
20	23.431	23.453	23.349
25	26.662	26.663	26.775
30	29.945	29.899	29.937

経験則からの評価であるが，実験者が NetEm を 1 台 1 台設定することを考えると，実験ネットワークの構築時間は日オーダーかかり，NetEm の設定に詳しい実験者であっても，かなりの時間がかかる．実験者が手作業で NetEm を設定することに比べて，本システムでは，30 本のリンク特性を模倣する設定を反映させるのに 30 秒弱の時間で構築できている．これは NetEm ノードが接続しているネットワーク機器への設定も含めての時間であるため，かなりの時間短縮に成功していると言える．

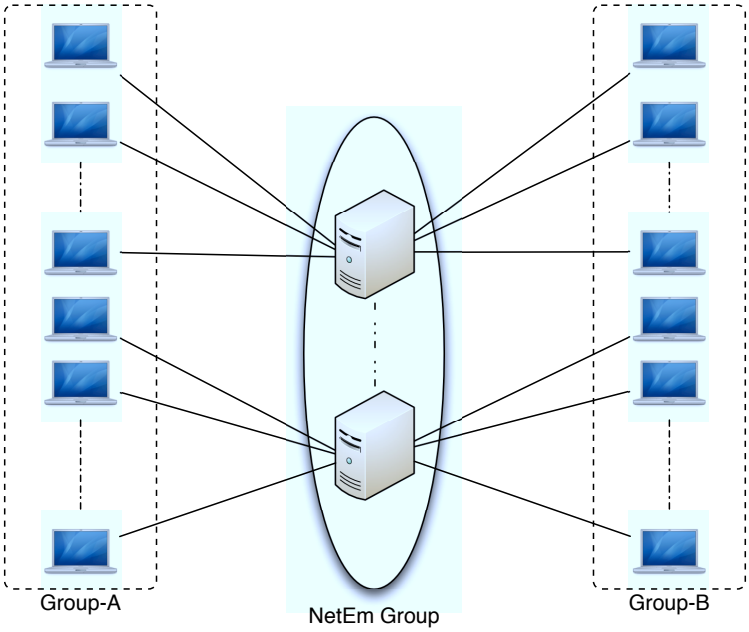

図 6.1: リンク特性を模倣した実験ネットワーク

本システムでは，実験者から与えられたインターネットの特性の値によってノードに要求されるリソースを計算し，適切に配置するため実験者が NetEm の詳細な設定を行う必要がない．そのため，実験者が NetEm を扱ったことのない場合においても，入力する設定ファイルの記述に時間はかからないと考えられる．

6.2 関連研究との比較

ネットワークの特性を模倣するシステムを提案している研究は本研究だけではない．そこで本項では，関連研究として挙げたシステムの中から，エミュレーションという形でネットワークの特性を模倣したネットワークを構築するシステムとの比較と行う．関連研究として Modelnet と SpringOS に含まれている K 言語による模倣ネットワークを比較対象とする．比較を行った結果を表 6.2 に示す．

6.2.1 実験ネットワークの構築に必要なノード数の比較

実証環境では，利用可能なリソースは限られている．実験者は利用可能なリソースの中で検証環境を構築し，検証を行う．インターネットの特性を模倣した検証環境を構築するためには，実証環境のリソースが必要である．そのため，インターネットの特性を模倣した検証環境の構築には，可能な限り少ないノード数で構築可能である方が，検証に実験

ノード数を多く利用できる．ここでは，インターネットの特性を模倣した実験ネットワークの構築に必要なノード数を比較する．

本システムでは，ノードが利用できるメモリや，ネットワークインターフェイスのリンクスピードを考慮し展開するため，実験ネットワークの構築に必要なノード数はインターネットの特性の模倣に必要なリソースに依存する．しかし，ノードが持つリソースに余裕がある限り，可能な限り少ないノードでインターネットの特性の模倣を行うため，SpringOS や Modelnet と比較すると少ないノード数で同じ規模の実験ネットワークを構築することが可能である．そのため，他のシステムと比較し，規模耐性は高いと言える．

SpringOS では，1 リンク 1 ノード用いるため，ノードの消費が激しく，大規模な実験ネットワークへの適用は難しい．これは，SpringOS がリソースをノード単位として扱っているためであり，ノード自体が持つリソースを使い切る実験を前提として作られていないためである．

Modelnet では，Virtual Node を 1 台のノード上で多重化し大規模なトポロジを構築するため，本システムや SpringOS と比較しオーバーヘッドが大きいと言える．また，コアネットワークと呼ばれるインターネットの特性を模倣する箇所で、多数のパスに対しリンク特性をエミュレートするため、実証環境で提供されているノードを多数用いての大規模な実験ネットワークの構築は難しい．

6.2.2 模倣可能なインターネットの特性の比較

実験者が想定するインターネット環境は多様である．本研究では，インターネットの特性を遅延，帯域，遅延揺らぎ，パケットロス，IP ルーティングと定義した．関連研究では，本研究と違う定義をしているため，ここでは各システムが模倣可能な特性を比較する．

本システムでは，NetEm を用いたネットワークエミュレートに対応しているが，dummynet や NIST Net などのネットワーク特性エミュレータには対応していない．そのため，模倣できるインターネットの特性は NetEm が扱える範囲に限定される．その一方，NetEm のブリッジ機能を用いたリンク特性の模倣と IP フォワーディングを用いたネットワーク特性の模倣の 2 種類の実験ネットワークを提供している．リンク特性の模倣では，IP ルーティングを行うソフトウェアの検証や，1 対 1 のみの通信の検証に利用でき，ネットワーク特性の模倣では，IP ルーティングを行わないソフトウェアや，1 対多通信を行うソフトウェアの検証に利用できる．本システムでは，ノード間のホップ数は模倣しない．これは，ノード間に存在する中継機器の処理時間の変化による遅延の揺らぎなどをすべて遅延揺らぎとして模倣できると考えたためである．

SpringOS では，NetEm と dummynet に対応しているため，利用可能なネットワーク特性エミュレータは状況に合わせて変更できる．しかし，SpringOS によるインターネットの特性の模倣では，ネットワーク特性エミュレータが IP フォワーディングを行ってのネットワーク構築機能しか提供していないため，同じネットワーク内の遅延や帯域を模倣することはできない．そのため，IP ルーティングを行うソフトウェアの検証を行う場合

には，ソフトウェアと実験ネットワーク側の両方でIPルーティングが行われることになるため，正しい検証を行えない．

Modelnetでは，dummynetのみの対応であるが，Virtual Nodeを用いることによってホップ数の模倣も可能としている．このホップ数の模倣は本システム，SpringOSにはない機能であり，模倣可能なインターネットの特性は豊富である．しかし，dummynetは遅延揺らぎの模倣機能を持っていない．dummynetで遅延揺らぎを模倣する場合には，複数の遅延生成を行い，パケットを確率的に適用されることで擬似的に模倣する．Modelnetでは，遅延揺らぎの模倣が擬似的であるが，多くの特性の模倣が行える．そのため，システムが模倣可能な特性の数は豊富である．

6.2.3 性能測定への利用

本研究では，ネットワーク技術を利用したアプリケーションや製品の検証を行う際に，インターネットの特性を模倣した実験ネットワークの構築が必要であると述べた．ここでは，インターネットの特性を模倣した実験ネットワークが検証に利用可能であるかどうかの比較を行う．

本システムでは，NetEmのアルゴリズムに基づきリソースの算出を行い，正しく模倣可能な限界まで模倣している．NetEmが必要とするメモリ量を計算し各NetEmノードへ割り当てているため，他のSpringOSやModelnetと比較し同じノード数でも大規模な実験ネットワークを構築することが可能である．本システムは，NetEmを多重化することによって実験ネットワークを構築しているが，NetEm自体は実ノードを用いている．そのため，仮想OSを用いた場合に起こる，実ノードとの同一性は考慮しなくてもよい．

SpringOSは1リンク1ノード用いているが，NetEm，dummynetを用いてネットワークの模倣を行っている．そのため，1本のリンクを模倣するために必要なリソースがノードの利用可能なリソースを超えていなければ，ネットワーク特性エミュレータがインターネットの特性を正しく模倣できる．SpringOSでの実験ネットワークにおいても，実ノードを用いているため，仮想OSと実ノードとの同一性は考慮しなくてもよい．

Modelnetでは，Virtual Nodeを用いてい実験ネットワークを構築する．そのため，ノードの物理的な特性の検証や，物理的な特性に関するソフトウェアなどは利用できない．このため，Modelnetで構築した実験ネットワーク上で性能測定用途への利用は難しい．またVirtual Node自体が実ノードとどの程度の同一性を持っているかの検証はされていない．実験者は，検証を行う前にModelnetで利用するVirtual Nodeが実ノードとどの程度の同一性を持っているかを調査しておく必要がある．

6.2.4 実験シナリオとの同時実行

実証環境上で利用する際に，実験シナリオと同時利用可能であるかを考える．本システムでは，SpringOSに含まれているERMとSWMGを利用しているが，基本的に実験シ

ナリオに記述する必要はない．しかし，実験シナリオ内に本システムを用いるよう記述していれば，実験シナリオ実行時に模倣ネットワークの展開が可能である．本システムは，実験シナリオ内で実験ネットワークの構築を行うよう設計はしていないが，実験シナリオ内で本システムを使用するよう記述していれば，実験シナリオ実行時に実験ネットワークを構築し実験を行うことが可能である．そのため，実験ネットワークを構築した上でシナリオを実行するといった手順は必要なく，実験シナリオ実行と同時にインターネットの特性を模倣したネットワークを構築することが可能である．

SpringOS ではそもそも模倣ネットワークの展開を実験シナリオの記述に使用する K 言語で記述されるため，実験シナリオを同じように模倣ネットワークの構築が可能である．実験シナリオのための K 言語で記述されることから，実験者は実験シナリオの中にインターネットの特性を模倣したネットワークの構築手順を記述する必要がある．そのため，K 言語の記述方法さえ習得していれば，自由に実験ネットワークを構築することができる．その一方，K 言語による実験シナリオを用いない場合には，K 言語で実験ネットワークの構築手順を記述した上で，実験を行う必要がある．

Modelnet では，ネットワークの構築時にエッジノードと呼ばれるアプリケーションを実行する可能ノードが作成されるが，実験シナリオを用いての使用は考慮されていないため，Modelnet 単体に実験シナリオ実行機能はない．また，仮想ノードを用いるため，Modelnet を用いてのネットワーク上で他の実証環境で提供されているシナリオ実行を行うのは難しい．

6.2.5 他の実証環境との協調性

本研究で，実証環境と定義しているのは StarBED だけではない．StarBED の他にも PlanetLab や GARIT，Netbed など様々な実証環境が存在する．ここでは，様々な実証環境とインターネットの特性を模倣するシステムとの協調性を比較する．

本システムでは，SpringOS に含まれる ERM や SWMG を利用する．そのため，ERM が実験設備の情報を把握していることと，ネットワーク機器を実験者が設定可能であることが条件となる．この条件は，PlanetLab のようなインターネットを利用している実証環境では，満たすことができない．したがって，本システムを他の実証環境に導入する場合，ERM と SWMG が動作する環境である必要がある．しかし，AnyBED のようなインターネットから隔離された環境かつ，ネットワーク機器を実験者が設定可能であれば，本システムを用いて実験ネットワークを構築することができる．

SpringOS は，ハードウェア的制限は特に存在しない．そのため，ERM が動作していれば，SpringOS によるインターネットの特性の模倣は可能である．また，PlanetLab のようなインターネットを利用した実証環境との協調性では，松井ら [19] の研究によって，StarBED と PlanetLab との連携が行われている．したがって，他の実証環境との協調性は高いと言える．

Modelnet は，Virtual Node を用いているため，PlanetLab のような仮想環境を用いて

いる実証環境では，導入が難しい．一方，実証環境として，仮想環境を用いていないのであれば，Modelnet の制限は特に存在しない．そのため，実証環境の運用ポリシーによって制限がかかると言える．

表 6.2: 機能比較

システム	提案システム	SpringOS	Modelnet
必要なノード数	○	×	△
模倣可能な特性	○	○	○
性能測定への利用	○	○	△
実験シナリオとの同時実行	△	○	×
他の実証環とでの協調性	△	○	△

第7章　おわりに

本稿では，実証環境上で検証を行う際に必要となるインターネットの特性について，その必要性や問題について述べてきた．実証環境で利用可能なリソースの中で，可能な限り大規模な実験ネットワークを構築するための方法として，実証環境上のノードを用いてインターネットの特性を模倣することが可能なネットワーク特性エミュレータに注目した．ネットワーク特性エミュレータの多重化によるリソースの効率的な利用と，自動的に構築可能な実験ネットワークの構築手法についての提案を行った．提案システムでは，インターネットの特性を遅延，帯域，パケットロスなどのノード間のリンクで発生するリンク特性と，複数のネットワーク機器の挟んだネットワーク上でノード間の経路を作成するIP ルーティングを含んだネットワーク特性に分類した．また，ノードが持つメモリやネットワークインターフェイスなどのリソースの考慮しつつ，リンク特性，ネットワーク特性を模倣した実験ネットワークの構築手法を提案した．そして，リンク特性を模倣する実験ネットワークとネットワーク特性を模倣する実験ネットワークを自動的に構築するシステムの実装を行った．

提案したインターネットの特性を模倣した実験ネットワークの構築方法の実験を行った．そして，インターネットの特性に注目した関連研究として，SpringOS に含まれている K 言語と Modelnet との比較を行い，それぞれの利点，欠点に関して考察を行った．

今後の課題

本研究で，提案した手法，システムでよりよい検証を行うための実験ネットワークを構築するために以下のような方法が考えられる．

1. 動的なインターネットの特性の変更
   本研究の提案システムは，実験者がインターネットの特性として，遅延や帯域などを固定値として扱っていた．これらの値を実験中に動的に変更するシステムを作成することにより，時間によるインターネットの特性の変化を再現できるようになる．これらの再現はより現実的な実験ネットワークを構築することができる．

2. 複数の模倣形態を混合した実験ネットワークの構築
   本研究では，実験ネットワーク側で IP ルーティングを提供するものと，IP ルーティングを提供しないものの 2 種類の実験ネットワーク構築手法を提案した．しかし，あ

る部分はIPルーティングを行い，その他では行わないとった2つの実験ネットワークを混合した実験ネットワークであれば，さらに柔軟な検証が可能であると考えられる．

3. より現実的な帯域制限手法
   本研究で利用したトークンバケツフィルタは，一定速度を超えたパケットを破棄するテイルドロップ型の帯域制限手法である．テイルドロップ型の帯域制限手法では，利用可能な帯域幅の上限を完全に決めてしまう．実際のインターネットでは，時間単位で見たときにある帯域を測定できたとしても，それは一定の帯域幅を使い続けていたわけではない．また，観測される帯域の値を実験ネットワークに模倣した場合，上限が設定される帯域幅をすべて使い続ける必要がある．したがって，実験ネットワークを流れるトラフィック量が平滑化されたときに設定された値と同じになるような帯域制限手法を利用することにより，よりよい実験ネットワークの構築が可能であると考えられる．

4. 無線環境の模倣
   本研究では，有線環境を対象に実験ネットワークの構築を進めた．しかし，インターネットは多様化しており無線ネットワークを利用したアプリケーションや製品も増えている．有線環境と無線環境では，構築する実験ネットワークは変化する．そのため，無線環境で観測される特性を模倣する実験ネットワークの構築システムが必要である．

謝辞

本研究を進めるにあたり，多くの方にお時間を頂き，多大なご指導を頂きました．その方々のご指導，ご助言がなければ，本研究は成り立ちませんでした．ここに心から深く御礼申し上げます．

本研究を進めるにあたり，指導教員である篠田 陽一 教授には，様々な助言，適切なご指導を賜りました．ここに心から深謝致します．主テーマ審査員である敷田 幹文 准教授，日比野 靖 教授，本学 丹 康雄教授，副テーマ指導教員である宮地 充子 教授には本研究に関して，貴重なご意見，ご指導を賜りました．ここに深く感謝いたします．

本研究室の知念 賢一 特任准教授，宇多 仁 助教，小原 泰弘 助教には，研究に関して多大なご指導を賜りました．また，研究環境構築研究，実験等様々な方面に関して多大なご指導を賜りました．ここに深く感謝いたします．

本研究を行うにあたり，Panasonic 株式会社 村本 衛一氏には，プロポーザルの時期より適切な助言やご指導頂けましたことをここに心から深く感謝いたします．

情報通信研究機構北陸リサーチセンターの三輪 信介氏，宮地 利幸氏，太田 悟史氏，中井 浩氏，佐野 正行氏，竹中 ゆかり氏，石崎 淳氏には StarBED での実験において多大なご助力頂けましたこと心より感謝いたします．

本研究室の博士後期課程の高野 祐輝氏，LATT Khin Thida 氏，井上 朋哉氏，安田 真悟氏，Nguyen Lan Tien 氏，芳炭 将氏，NGUYEN Nam Hoai 氏，Muhammad Imran Tariq 氏には，研究に関して活発な議論，ご指導頂くことができました．特に，高野氏，井上氏，安田氏には，本論文の研究方針について多大なご協力を頂きました．ここに心から深く感謝いたします．

本研究室の博士前期課程の松井 大輔氏，佐川 喜昭氏，石渡 優祐氏，栗原 良尚氏，川瀬 拓哉氏，立花 一樹氏，中村 祐輔氏，橋本 将彦氏，山田 悠介氏，吉岡 慎一郎氏には，研究に関して活発な議論を頂き，また有意義な研究生活を送ることができました．特に，松井氏，佐川氏，栗原氏には活発な議論や，研究生活を送る上で様々なご助力を頂きました．ここに心から深く感謝いたします．

最後に，研究や生活を支えてくれた家族へ心から感謝致します．

参考文献

[1] 梅木孝志. マルチレベル背景トラフィック生成技術に関する研究. Master's thesis, 北陸先端科学技術大学院大学, 2008.

[2] Y. Rekhter. A border gateway protocol 4 (bgp-4). *RFC 1771*, March 1995.

[3] John T. Moy, editor. 詳解 OSPF. 翔泳社. 小原 泰弘 監修.

[4] Toshiyuki Miyachi, Ken ichi Chinen, and Yoichi Shinoda. Starbed and springos large-scale general purpose network testbed and supporting software. *International Conference on Performance Evaluation Methodlogies and Tools (Valuetools) 2006*, Oct 2006.

[5] Brian White, Shashi Guruprasad, Mac Newbold, Jay Lepreau Leigh Stoller, Robert Ricci, Chad Barb, Mike Hibler, and Abhijeet Joglekar. Netbed: an integrated experimental environment. *ACM SIGCOMM Computer Communications Review*, Vol. 33, p. 27, July 2002.

[6] Brent Chun, David Culler, Timothy Roscoe, Andy Bavier, Larry Peterson, Mike Wawrzoniak, and Mic Bowman. Planetlab: An overlay testbed for broad-coverage services. *ACM SIGCOMM Computer Communication*, Vol. 33, pp. 2–13, July 2003.

[7] Mio Suzuki. Anybed. http://sourceforge.net/projects/anybed/.

[8] Daniel Mahrenholz and Svilen Ivanov. Real-time network emulation with ns-2. *Eighth IEEE International Symposium on Distributed Simulation and Real-Time Applications*, pp. 29–36, 2004.

[9] 宮地利幸. 大規模実証環境の実現と実験支援によるネットワークサービスの検証技術. PhD thesis, 北陸先端科学技術大学院大学, March 2007.

[10] Philip Lieberman. Wake on lan technology, July 2002.

[11] Mark Carson and Darrin Santay. Nist net - a linux-based network emulation tool. *ACM SIG- COMM Computer Communications Review*, Vol. 33, No. 3, pp. 111–126, July 2003.

[12] LuigiRizzo. Dummynet:asimpleapproachtothe evaluation of network protocols. *ACM SIGCOMM Computer Communication Review*, Vol. 27, No. 1, pp. 31–41, Jan 1997.

[13] S Hemminger. Network emulation with netem. *Linux Conf*, April 2005.

[14] iproute2. http://linux-net.osdl.org/index.php/iproute2/.

[15] Pramodsanaga, JonathonDuering, RobertRicci, and Jay Lepreau. Modeling and emulation of internet paths. *Sixth USENIX Symposium on Networked Systems Design and Implementation (NSDI)*, pp. 199–212, Apr 2009.

[16] Shinsuke Miwa, Mio Suzuki, Hiroaki Hazeyama, Satoshi Uda, Toshiyuki Miyachi, Youki Kadobayashi, and Yoichi Shinoda. Experiences in emulating 10k as topology with massive vm multiplexing. *In Proceedings of The First ACM SIGCOMM Workshop on Virtualized Infrastructure Systems and Architectures(VISA 2009)*, August 2009.

[17] Tatu Ylonen. Secure login connections over the internet. *Sixth USENIX Security Symposium*, pp. 37–42, March 1996.

[18] paramiko - ssh2 protocol for python. http://www.lag.net/paramiko/.

[19] 松井大輔, 櫻井覚, 管文鋭, 今井祐二, 村本衛一, 河口信夫, 篠田陽一. 新広域サービスの段階的検証手法への planetlab の適用. *DICOMO 2007*, 2007.

本研究に関する発表論文

- 明石邦夫, 井上朋也, 安田真悟, 村本衛一, 知念賢一, 宇多仁, 篠田陽一. リンクエミュレータの多重実行の限界値測定. Internet Conference 2009, pp. 33—39, Oct 2009.

- Thilmee Baduge, Boon Ping Lim, Kunio Akashi, Jason Yew Beng, Ken ichi Chinen, Ettikan Kandasamy Karuppiah, and Eiichi Muramoto. Functional and performance verification of overlay multicast applications - a product level approach. IEEE Consumer Communications and Networking Conference 2010, Jan 2010.