卒業論文 2015 年度 (平成 27 年)

Bootstrap に向けた Swift による Swift 構文解析器の設計と実装

慶應義塾大学 環境情報学部
出水 厚輝

卒業論文要旨 - 2015 年度 (平成 27 年度)

Bootstrap に向けた Swift による Swift 構文解析器の設計と実装

現在利用されている多くの高級な汎用プログラミング言語では、コンパイル対象となる言語自体でそのコンパイラを記述する Bootstrap が行われている。Bootstrap を行うことによるメリットはいくつかあるが、度々モチベーションとしてあげられるのは、現存するプログラミング言語よりも Bootstrap を行おうと考えているプログラミング言語のほうが後発のものであるため、より表現力が高く開発しやすいという点である。

しかし、近年開発されている汎用プログラミング言語に至っては、その言語自体だけでなく最初にコンパイラを記述する言語も高級なものとなっており、対象のコンパイラを記述する上でどちらの方がより高い表現力や性能を持つかを簡単に判断することはできなくなってきている。

Apple 社が中心となって開発しているプログラミング言語 Swift もそのメリットとデメリットを明確に評価することができず、Bootstrap するべきか否かの判断を下せていない汎用プログラミング言語の 1 つである。現在最も有名な Swift のコンパイラ実装は C++ で記述されており、コンパイラの核となる構文解析においても C++の特徴的な機能を駆使して、より低級な言語ではボイラープレートとなるコードを排除している。Swift はその可読性の高さと実行速度の速さを謳った言語であるが、その性能が Swift コンパイラという大規模なソフトウェアにおいて C++を相手としても通用するものであるかどうかを形式的に議論することは容易ではない。

そこで本研究では、Swift で記述した Swift の構文解析器を実装し、その実行時間とソースコードの構文的特徴を現行の Swift コンパイラ中の構文解析器と比較することで、Swift が Bootstrap を行うための判断材料を収集・考察する。本論文では、Swift で構文解析器を書き換えることによって可読性につながりうるソースコードの行数の削減は実現できるが、実行速度の面においては未だ Swift 自体が充分な性能を持っていない可能性があることを示し、その結果から Swift が Bootstrap を行うならば必要になるであろうステップについて考察を行っている。

キーワード:
1. コンパイラ・ブートストラップ, 2. 構文解析, 3. 構文解析器の実装,
4. プログラミング言語 Swift

慶應義塾大学 環境情報学部

出水 厚輝

Abstract of Bachelor's Thesis - Academic Year 2015

Design and Implementation of Swift Parser Written in Swift for Bootstrapping

English summary here.

Keywords :
1. Bootstrap a Compiler, 2. Parser, 3. Implementation of Parser,
4. Swift Programming Language

Keio University, Faculty of Environment and Information Studies

Atsuki Demizu

目 次

図 目 次

表 目 次

第1章　序論

インターネット上のサービスを提供するサーバー群は、サービスレスポンス時間の短縮やサービス冗長化のために、複数のデータセンター拠点に展開されている。インターネット上のサービスは複数のコンポーネントによって構成される。物理的に離れた複数の拠点を用いてサービスを構築するには、他の拠点に設置されたコンポーネントを利用する必要のある場合がある。物理的に離れた拠点に設置されたコンポーネントを利用すると、拠点間の遅延がコンポーネントのパフォーマンスに大きく影響を与える。しかし、物理的に離れた拠点間がインターネットを経由して通信する際の経路は、宛先までの遅延や回線の使用率などを考慮して選択された経路ではないため、遅延が大きくなってしまう場合がある。

1.1　拠点の複数化

近年、インターネット上のサービスはサービスのレスポンス時間を小さくするためや、自然災害やそれに伴う停電などによるサービスの停止を避けるため、サービスを複数の拠点に設置している場合が多い。以下で、拠点が複数化する理由の詳細を述べる。

1.1.1　レスポンス時間の改善

サービスの利用者がそのサービスに対しての満足度は、そのサービスのレスポンス時間と関係している。2001 年に Zone Research 社が行った調査によると、当時、インターネット上のサービスを利用するにあたり、そのサービスに満足できるレスポンス時間は 8 秒であった [1]。レスポンス時間が 8 秒以上であった場合、そのサービスを利用しなくなってしまうという結果も出ている。そして、2006 年に同じ調査を Akamai 社が行った際の結果では、インターネット上のサービスに求められるレスポンス時間は 4 秒という結果となっ

表 1.1: サービスに求められるレスポンス時間

年	求められるレスポンス時間
2001 年	8 秒
2006 年	4 秒
2009 年	2 秒

た [2]。更に、同社が 2009 年に同じ調査を行った際には 2 秒と更に短くなっていた [3]。これを表 1.1 に示した。今後、利用者から求められるレスポンス時間は更に短くなることが予想される。そのため、サービスのレスポンス時間が小さいほど、利用者は満足すると言える。

近年、サービスを運用するに当たり、サービスの利用者が世界中のどこからでもサービスを満足して利用できるよう、世界中に設置されたサーバー群の中で最も利用者が満足して利用できるサーバーからサービスを提供するという手法が用いられることが多い。この様な手法でサービスを運用しているサービスの例として、Akamai 社 [4] が提供している Content Delivery Network(=CDN) が挙げられる。Akamai 社は 2010 年の時点で 61,000 台のサーバーを、1,000 以上のネットワーク、世界 70 カ国以上の拠点に持っている [3]。サービスの利用者が Akamai 社のサービスを利用する際には、このサーバー群の中から、利用者のネットワークまでの遅延や経路などといった情報をもとに、利用者にとって最も低遅延でサービスを提供できるサーバーを選択する。これによって利用者は世界中のどこからサービスを利用しても、満足してそのサービスを利用できるようになる。

このように複数の拠点から同じサービスを提供するという手法は、レスポンス時間を最小限にする手法として非常に有効である。単一の拠点からサービスを提供した場合、サービスを提供している拠点とそのサービスの利用者が離れているとレスポンス時間は長くなる。多くの拠点で同じサービスを提供した場合、利用者にとって一番近い拠点からサービスを行うことができるので、単一の拠点から提供した場合と比べるとレスポンス時間は短くなる。レスポンス時間を短くするために、このようなシステムを運用するこサービスが増えてきているため、サービスを提供するための拠点が増加している。

1.1.2 自然災害によるサービス停止の防止

自然災害やそれに伴う停電などにより、データセンターが完全に停止してしまう可能性がある。例えば、2011 年 3 月 11 日に起きた東日本大震災では、多くのインターネット上のサービスが一時的に停止した。震災直後は、日本の多くのデータセンターは震災の影響を受けることなく、サービスは継続した。しかし、その後関東圏で電力不足が生じ、計画停電が実施された。この影響により、NTT コミュニケーションズ社 [5] が提供している一部法人向けサービスが利用できなくなった [6]。慶應義塾大学湘南藤沢キャンパス [7] でも計画停電が実施され、多くのサービスを停止させる必要があった。また、海外でも同様に自然災害によりサービスが停止してしまうという事例がある。2012 年 10 月、米国東部にハリケーンが上陸し、ニューヨークでは広範囲で浸水に見舞われた。多くの企業が利用している DataGram 社 [8] のデータセンターでは、地下が浸水してしまい、その影響により送電網が利用できなくなった。これにより建物全体が停電し、完全に復旧するのに 1 週間近くかかった [9]。また、Internap 社 [10] や PEER 1 Hosting 社 [11] のデータセンターも同様に停電し、全てのサービスが停止した [12]。

インターネット上のサービスが一時的に利用できなくなると、そのサービスを利用しなくなってしまう場合がある。2009 年に行われた調査によると、有名なオンラインショッ

ピングサイトが 1 時間利用できなくなった場合、損失は 280 万ドル以上になるという [3]。このような損失を防ぐためには、サービスの停止を防ぐ必要がある。

自然災害や障害によるサービス停止を防ぐために、近年では複数拠点から同じサービスの提供をすることや、サービスの複製を行うことが多い。近年では、インターネット上のサービスの多くが、単一の拠点内で構成される冗長構成だけでなく、複数の拠点を利用した何重もの冗長構成で構築されている。複数の拠点を利用した冗長構成では、サービスを提供している拠点の 1 つが利用できなくなった場合でもサービスの継続が可能である。例えば、Amazon Web Services 社 [13] が提供している S3 ストレージサービスは、複数の拠点に同じデータを複製している [14]。そのため、1 つの拠点が利用できなくなった場合でも、他拠点から同じデータを利用することができる。このような構成で行われるサービスが増えているため、サービスを提供するための拠点が増加している。

1.2 遅延が考慮されないインターネットの経路

現在のインターネットの経路は必ずしも遅延が最も小さい経路ではない。インターネットの経路は Border Gateway Protocol(=BGP) によって学習と選択がされている。BGP では経路が最もホップする Autonomous System(=AS) の数が少ないものを優先して経路が選択される。直接経路を交換している (=ピアリング) を行なっている相手と通信する際には、ピアリングを行なっている拠点を経由する。ピアリングを行なっていない相手に関してはトランジットを経由する。この仕組では、例えば、通信元と通信先が両方東京に設置されていたとしても、通信元のネットワークと通信先のネットワークが大阪でピアリングを行なっている場合、通信は必ず大阪を経由する。このように BGP を利用した経路選択は、宛先に到達するまでの遅延や距離、リンクの占有率などを考慮していない。また、BGP にはビジネスの側面がある。例えば、2008 年 10 月 30 日、アメリカの大手インターネットプロバイダーである Cogent Communications 社 [15] と Sprint 社 [16] 間でピアリングを行うにあたっての条件を両者で合意できなくなったため、ピアリングを止めた [17]。これにより、両者のネットワーク間で通信するためには、非常に遠い経路を通る必要が生じた。このように、BGP を利用した現在のインターネットの経路では、宛先によっては遅延の大きい経路が選択されてしまう可能性がある。

遅延を削減するためには、他の拠点を一旦経由してから最終的な宛先に到達することで BGP による経路よりも小さい遅延で宛先に到達できる場合がある [18]。MIT の Hariharan Rahul 氏や Arthur Berger 氏らによる研究 [19] では、世界 77 カ国からなる 1100 以上の拠点に設置されたサーバー間の遅延を計測した。その結果、世界中の全経路のうち約 35%の経路が、他の拠点を経由することによって 30%の遅延削減が可能だ、とされている。特に通信元と通信先の両方がアジア圏の通信の一部では、他の拠点を経由することにより、50%以上の遅延削減が可能である場合もあった。このように、現在のインターネットではインターネットの経路で直接転送するよりも、他の拠点を経由させることによって、より小さい遅延で通信できる場合がある、ということがわかる。

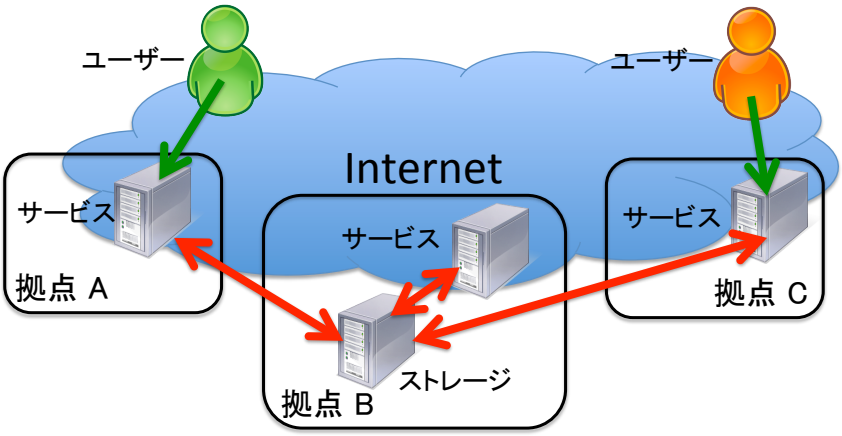

図 1.1: 複数の拠点から提供されるサービス

1.3 拠点の複数化による利点と欠点

インターネット上のサービスを複数の拠点で展開することにより、サービスのレスポンス時間の向上や耐障害性の向上などといった利点を得ることができる。しかし、その一方で、広域にサービスを構成するコンポーネントが分散することにより生じる遅延により、場合によってはレスポンス時間が低下してしまう可能性がある。以下で、このような拠点の複数化を行った際に生じる欠点を詳しく述べる。

1.3.1 Layer 2 ネットワーク拡張技術を用いた複数拠点の運用

利用者がサービスを利用するにあたり、利用者はサービスが提供するフロントエンドのインターフェースのみを参照する。しかし、インターネット上のサービスは、図 1.1 で示すように、サーバーやストレージ、ネットワークゲートウェイなどといった複数のコンポーネントから構成されている。フロントエンドのインターフェースの裏では、サービスを動かすために必要となるストレージやデータベースなどといった様々なバックエンドのコンポーネントが存在する。例えば、オンラインストレージサービスは、主に利用者が実際にアップロードやダウンロードなどの操作を行う Web サーバー、実際にアップロードされたファイルが記憶されているストレージサーバー、そしてファイルやユーザー情報の管理を行うためのデータベースサーバーといった 3 つのコンポーネントから成り立っている。この内、利用者が操作を行う部分は Web サーバーが提供するフロントエンドインターフェースのみである。残り 2 つのコンポーネントを利用者は関知しない。このようにインターネット上のサービスは、利用者が直接関知しない多くのコンポーネントから構成されており、全てのコンポーネントを連動させることによってサービスが成り立っている。

サービスを運用するにあたり、コンポーネント間には適切なセキュリティーポリシーを適用する必要がある。サービスを構成させるコンポーネントにはインターネットからアクセスを許可すると危険なコンポーネントがある。このようなコンポーネントの例として、

RPC [20] や Samba [21] など既知のセキュリティーホールが非常に多いものが挙げられる。また、利用者のパスワードやデータなどが記憶されていて侵入されると被害が大きいコンポーネントもある。このようなコンポーネントを利用してサービスを構成する際には、コンポーネント間のセキュリティーポリシーを正しく設定する必要がある。セキュリティーポリシーはコンポーネントによって異なる。インターネットからのアクセスを一切受け付けなくても良いコンポーネントもあれば、一部の通信のみ受け付ける必要があるコンポーネントもある。このように個々のコンポーネントに対して異なるセキュリティーポリシーを適用する必要がある場合は、複数の Layer 2 ネットワークを用いて、セキュリティーポリシーの適用を行う。

分散された複数の拠点においてサービスを展開する場合でも、適切にコンポーネント間のセキュリティーポリシーを適用する必要がある。しかし、複数の拠点においてコンポーネント間のセキュリティーポリシーを適用するために、拠点毎にコンポーネント間のセキュリティーポリシーを設定するのは、設定が複雑化しサービスを運用するコストが非常に高くなる、という問題がある。また、インターネットからのアクセスを許可すると危険なコンポーネントを他の拠点と共有するためには、そのコンポーネントをインターネット上に設置する必要があるため、危険が生じるという問題がある。これらの問題を防ぐために、Layer 2 ネットワーク拡張技術を用いて、Layer 2 ネットワークを全拠点に延長することにより、同じセキュリティーポリシーのもとでコンポーネント間の通信を可能にする。

しかし、広域な環境では 1 つの拠点内で運用した場合と比べて遅延が非常に大きいため、広域な環境ではコンポーネントのパフォーマンスが著しく低下してしまう場合がある。1 つの拠点内でのコンポーネント間の遅延は、コンポーネント間の距離が物理的に近いため遅延を考慮する必要がない。それに比べ、複数の拠点に分散されたコンポーネント間の遅延は、コンポーネント間の距離が物理的に離れている上に、インターネットを経由する。物理的に離れている場合、データを転送するために時間がかかるため遅延が発生する。さらに、インターネットでは拠点間が通信するために、多くの拠点を経由する。それにより更に遅延が発生する。また、経由する拠点にトラフィックが集中している場合は、加えて遅延が発生する。そのため、コンポーネント間の遅延は、1 つの拠点内と比べると非常に大きい。インターネット上のサービスを構成するために利用される技術の多くは 1 つの拠点内で利用されることが想定されていることが多い。そのため、インターネットのような遅延が大きい環境でコンポーネントを利用した場合、そのコンポーネントのパフォーマンスが著しく低下してしまう場合がある。

1.3.2 WIDE Cloud を例にした複数拠点運用における遅延の影響

広域環境で運用した際に、パフォーマンスが低下してしまうサービスの例として、WIDE Project [22] が運用する広域に分散された IaaS 型のクラウド環境である WIDE Cloud [23] が挙げられる。WIDE Cloud 上に利用者は、仮想マシンを自由に立ち上げることができる。そして、作成した仮想マシン上で、利用者は任意のサービスを自由に立ち上げることができる。実際、WIDE Cloud 上では多種多様なサービスが立ち上げられている。しか

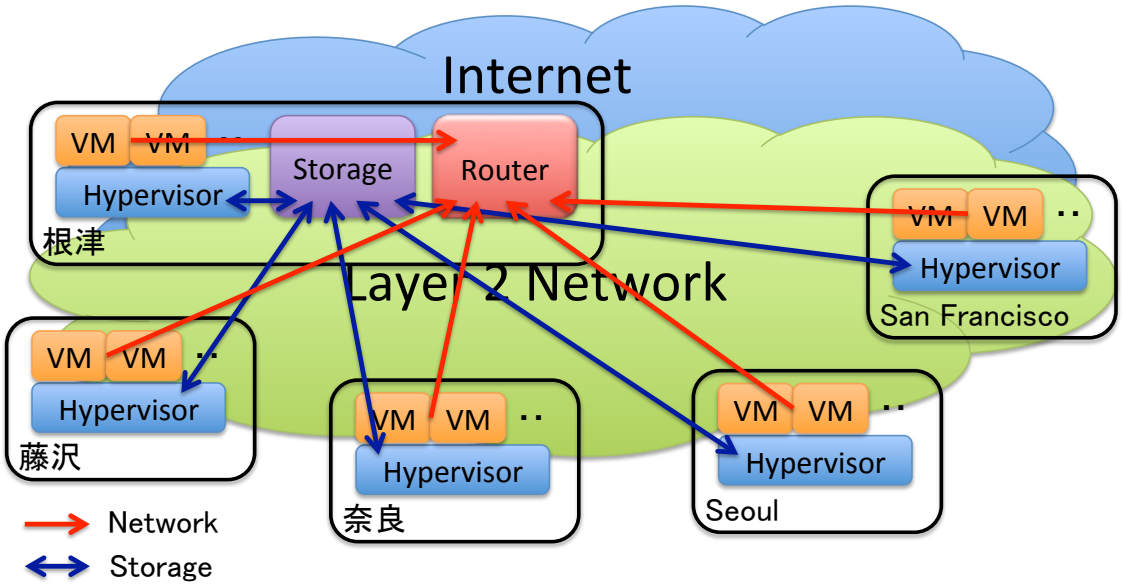

図 1.2: WIDE Cloud を構成するコンポーネント

し、これらのサービスのパフォーマンスは、WIDE Cloud のパフォーマンスに影響されている。

WIDE Cloud を構成するコンポーネントとして主に、ハイパーバイザー、ストレージとネットワークゲートウェイの 3 つのコンポーネントが挙げられる。各コンポーネントの関係を図 1.2 に示した。仮想マシン上でサービスを立ち上げるためには、CPU、メモリー、記憶領域とネットワークが必要である。この内、CPU とメモリーはハイパーバイザーによって提供される。そして、記憶領域はストレージによって提供される。最後に、ネットワークはネットワークゲートウェイによって提供される。これらのコンポーネントが連携することにより、仮想マシンが構成されている。

WIDE Cloud を構成するコンポーネントは、離れた複数の拠点に分散されているため、インターネットを経由して利用している。WIDE Cloud のハイパーバイザーは日本国内だけでなく、韓国やアメリカなどといった日本国外の拠点にも設置されている。しかし、ストレージとネットワークゲートウェイは日本国内の単一の拠点に設置されている。そのため、多くのハイパーバイザーは遠隔からストレージとネットワークゲートウェイを利用する。

拠点間でのコンポーネント通信を行うために、WIDE Cloud では、同一の Layer 2 ネットワークを全ての拠点に拡張している。Layer 2 ネットワークの拡張には、広域 Layer 2 ネットワークと Layer 2 ネットワーク拡張技術を利用している。各拠点に設置されたハイパーバイザーはインターネット上に拡張された Layer 2 ネットワークを経由して、ストレージとネットワークゲートウェイを利用している。

WIDE Cloud 上で動作する仮想マシンのパフォーマンス低下の原因は、ストレージとネットワークゲートウェイのパフォーマンス低下である。WIDE Cloud の多くのハイパーバイザーはストレージとネットワークゲートウェイをインターネット経由で利用する。ス

トレージとネットワークゲートウェイのパフォーマンスは遅延によって、パフォーマンスが著しく低下する。ストレージまでの遅延は仮想マシンのディスクパフォーマンスに影響する。ストレージをハイパーバイザーから利用するために、WIDE Cloud では NFSv3 [24] を利用している。しかし、既存の研究によると、iSCSI [25] と NFSv3 は遅延が大きくなるに連れ、パフォーマンスが低下する [26]。ストレージのパフォーマンスが低下すると、仮想マシンのディスクパフォーマンスが低下するため、仮想マシンのパフォーマンスが低下する。また、ネットワークゲートウェイまでの遅延は、仮想マシンのネットワークパフォーマンスに影響する。仮想マシンがインターネットと通信するためには、必ずネットワークゲートウェイを経由する。そのため、ネットワークゲートウェイまでの遅延が大きいほど、仮想マシンのネットワークパフォーマンスが低下する。このような仮想マシンのパフォーマンス低下は、コンポーネント間の遅延が小さいほど、パフォーマンスに与える影響が小さい。

コンポーネント間の遅延は、Layer 2 ネットワーク拡張技術がイーサネットフレームを転送する際に、拠点間の遅延を考慮することによって小さくすることができる。WIDE Cloud では、広域 Layer 2 ネットワークと Layer 2 ネットワーク拡張技術を用いて、全拠点に同一の Layer 2 ネットワークを拡張している。この 2 つの技術によって構築された Layer 2 ネットワークのトポロジーは木構造のトポロジーのため、一部拠点間の遅延が不必要に大きくなっている。また、現在のインターネットには 1.2 節で説明したような、直接宛先に転送するよりも、他の拠点を経由して転送するほうが、遅延が小さくなるような経路が存在する。Layer 2 ネットワークを拡張する際に、Layer 2 ネットワークを拡張する技術が遅延を考慮してイーサネットフレームを転送することにより、コンポーネント間の遅延が小さくなるため、遅延が仮想マシンのパフォーマンスに与える影響を小さくすることができる。

1.4 本研究の位置付け

本研究では、インターネット上に分散された複数の拠点に同一の Layer 2 ネットワークを拡張した際に、拡張された Layer 2 ネットワーク上で動作するサービスの遅延によるパフォーマンス低下を小さくすることを目的とする。これを実現するために、Layer 2 ネットワーク拡張技術がイーサネットフレームを転送する際に、遅延の最も小さい経路で転送することを可能にする。WIDE Cloud のように、サービスを構成するコンポーネントが複数の拠点に分散しているような構成では、コンポーネント間の遅延が、サービスのパフォーマンスに影響する。遅延が小さいほど、サービスのパフォーマンスは向上する。しかし、既存の Layer 2 ネットワーク拡張技術では、イーサネットフレームを転送する際に、遅延が最も小さい経路で転送されない。そのため、サービスのパフォーマンスが低下してしまっている。本研究では、拠点間の遅延を計測し、イーサネットフレームを遅延が最も小さくなる経路で転送する Layer 2 ネットワーク拡張技術を提案する。これにより、WIDE Cloud のように複数の拠点を用いて構築されたサービスのパフォーマンスを従来より向上させることができる。

1.5 本論文の構成

本論文の構成を以下に示す。第 2 章では、既存の Layer 2 ネットワーク拡張技術について整理し、分散した複数の拠点に Layer 2 ネットワークを拡張する際に求められる Layer 2 ネットワーク拡張技術を示す。第 3 章では、第 2 の内容をふまえた上で実現すべき機能要件について整理し、本研究で提案する手法について述べる。また、提案する手法の実装についても述べる。第 4 章では、行った実験とその実験から得られた評価結果を示す。そして、第 5 で、本研究のまとめと今後の展望を述べる。

第2章　関連研究

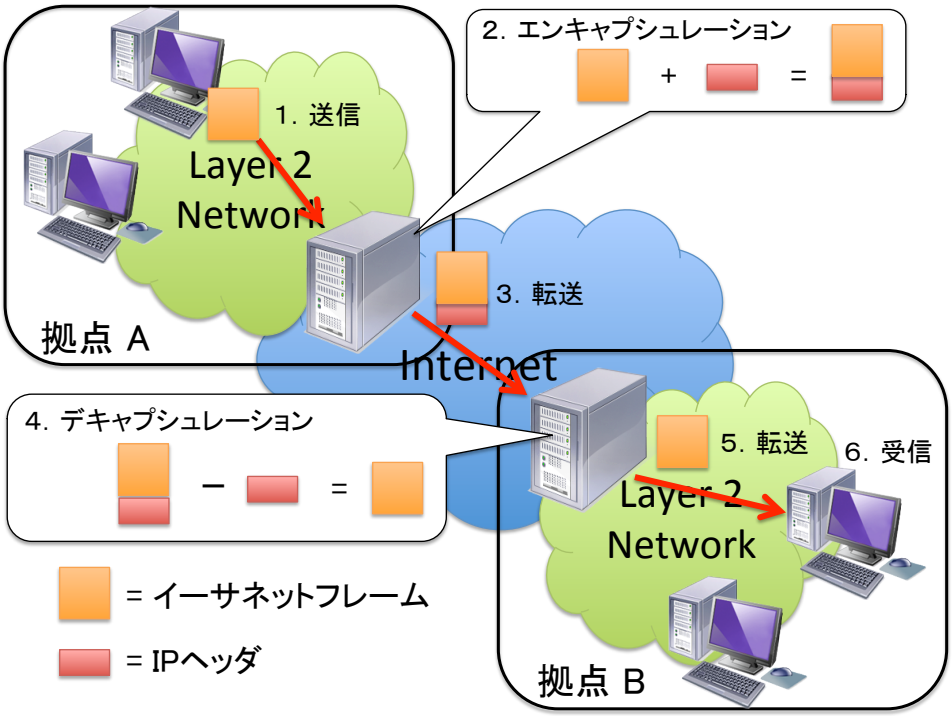

図 2.1: Layer 2 ネットワークの拡張手法

Layer 2 ネットワーク拡張技術を利用することにより、インターネット上に分散された複数の拠点に同一の Layer 2 ネットワークを拡張することができる。広域に分散した複数の拠点でサービスを展開するためには、サービスを構成しているコンポーネントを拠点間で共有する必要がある場合が多い。広域に分散した複数の拠点間で、共通のセキュリティーポリシーが適用された状態で、共通のコンポーネントを利用する手法の 1 つとして Layer 2 ネットワーク拡張技術を利用するという手法が挙げられる。Layer 2 ネットワーク拡張技術はインターネット上で Layer 2 ネットワークのイーサネットフレームを転送することにより、仮想的に広域な Layer 2 ネットワークを構築する。

Layer 2 ネットワーク拡張技術はトンネル終端点となるコンピューターがスイッチのように動作する。動作の仕組みを図 2.1 に示す。まず、Layer 2 ネットワーク拡張技術を利用して構築された Layer 2 ネットワーク上のホストが、トンネルの反対側にいるコンピューター宛のイーサネットフレームを送信する。それを受信したトンネル終端点は、受信した

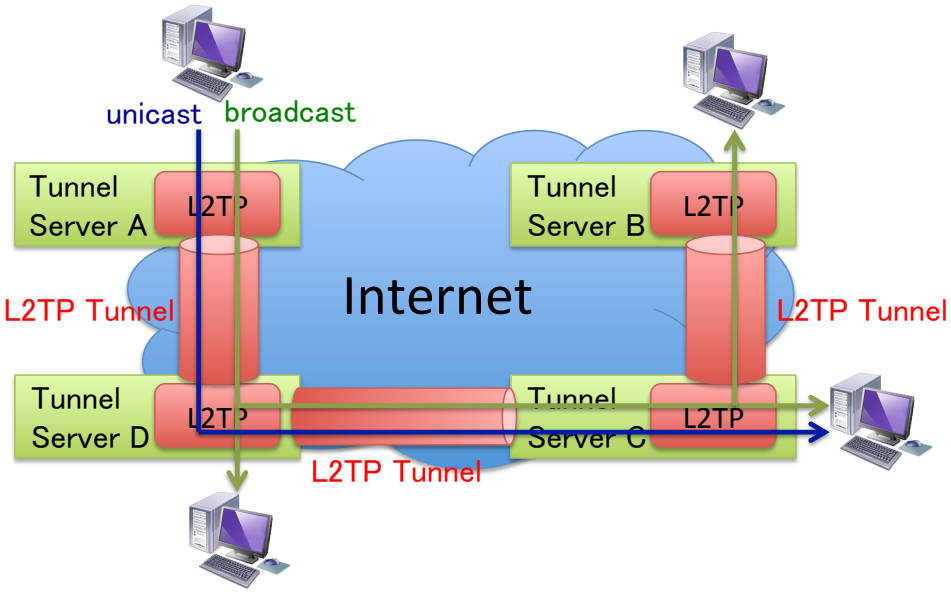

図 2.2: L2TP を利用して拡張された Layer 2 ネットワーク

イーサネットフレームの先頭に、宛先が反対側のトンネル終端点となっている IP ヘッダーを追加 (=エンキャプシュレーション) して送信する。そして、エンキャプシュレーションされたイーサネットフレームを受信したトンネル終端点は、追加された IP ヘッダーを取り除く (=デキャプシュレーション)。最後に、宛先のホストへイーサネットフレームを転送する。これにより Layer 2 ネットワークを拡張している。

既存の Layer 2 ネットワーク拡張技術は、一対一型の Layer 2 トンネル技術と一対多型の Layer 2 トンネル技術の 2 種類に分類することができる。以下で、それぞれの詳細を述べる。

2.1 一対一型の Layer 2 ネットワーク拡張技術

一対一型の Layer 2 ネットワーク技術は 2 つの拠点間で拡張された Layer 2 ネットワークを構築することができる。一対一型の Layer 2 ネットワーク拡張技術の例として、L2TP [27] や GRE [28] トンネルなどが挙げられる。L2TP を利用して Layer 2 ネットワークを拡張した際のトポロジーを図 2.2 に示す。一対一型の Layer 2 ネットワーク拡張技術では 2 拠点間で Layer 2 ネットワークを拡張する。3 拠点以上に同一の Layer 2 ネットワークを拡張するためには、一対一型の Layer 2 ネットワーク拡張技術を動作さえ、全ての拠点に同一の Layer 2 ネットワークを拡張する。そのため、図 2.2 で示したようなトポロジーとなる。しかし、この手法では一部の拠点間で行われる通信の遅延が大きくなってしまうという問題がある。

Layer 2 ネットワークのトポロジーは必ず木構造のトポロジーとなる。Layer 2 スイッチはブロードキャストやマルチキャストのイーサネットフレームを受信すると、そのイー

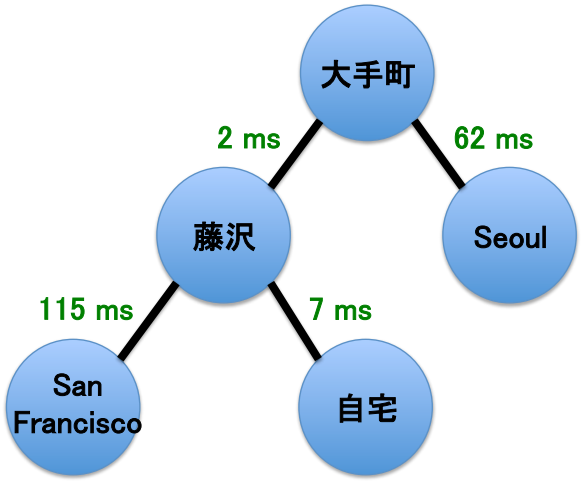

図 2.3: 木構造となる広域 Layer 2 ネットワークのトポロジー例 1

サネットフレームを受信したポート以外の全ポートにイーサネットフレームを転送する (=フラッディング)。フラッディングされたイーサネットフレームを受信した Layer 2 スイッチは同様ににそのイーサネットフレームをフラッディングする。そのため、リング型のトポロジーでは、ブロードキャストストームやマルチキャストストームが発生してしまうという問題 (=L2 ループ) が生じる。L2 ループが生じると帯域がブロードキャストやマルチキャストのイーサネットフレームで専有されてしまう上に、Layer 2 スイッチの CPU 負荷が高くなり正常にイーサネットフレームを転送できなくなる。その結果、Layer 2 スイッチに接続されているホスト同士も通信不能となってしまう。このような問題を防ぐために、Layer 2 ネットワークは木構造のトポロジーで構築される。また、Spanning Tree Protocol(=STP) を利用することで、L2 ループを発生させずにリング型のトポロジーを構築することも可能だが、STP は L2 ループを防ぐためにリングとなっているポートの通信を拒否する。その結果、STP を利用した場合でも、Layer 2 ネットワークのトポロジーは木構造のトポロジーとなってしまう。

一対一型の Layer 2 ネットワーク拡張技術を利用し、Layer 2 ネットワークを複数の拠点に拡張した場合も同様に木構造のトポロジーとなる。単一の拠点内で構築される Layer 2 ネットワークの場合、Layer 2 スイッチ間の遅延はほぼ無い状態なので、木構造のトポロジーでも問題とならない。しかし、Layer 2 ネットワーク拡張技術を利用してインターネット上に構築した Layer 2 ネットワークでは、1 ホップあたりの遅延が、単一の拠点内で構築した Layer 2 ネットワークと比べ、非常に大きい。例えば、図 2.3 で示すような広域な Layer 2 ネットワークでは、自宅に設置されたホストとソウルに設置されたホスト間で通信を行った場合、その遅延は 71 ms となる。図 2.4 で示すように、自宅とソウルを直接接続した場合、遅延は 35 ms となる。しかし、藤沢や大手町に設置されたホスト間で通信を行った場合の遅延が 97 ms 以上となってしまう。このように、一対一型の Layer 2 ネットワーク拡張技術では、Layer 2 ネットワークのトポロジーが木構造となってしまうため、必ず一部拠点間の遅延が大きくなってしまうという問題がある。

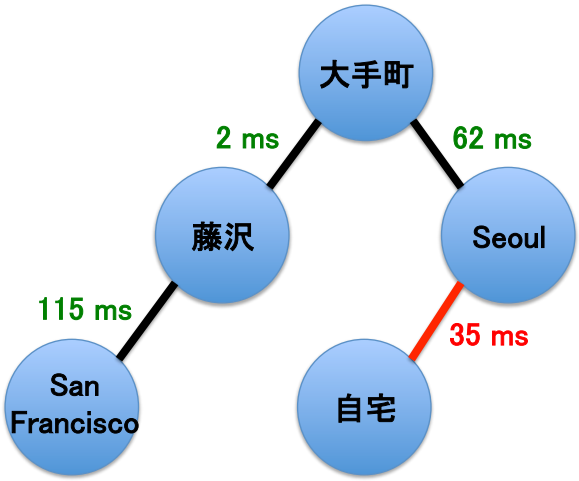

図 2.4: 木構造となる広域 Layer 2 ネットワークのトポロジー例 2

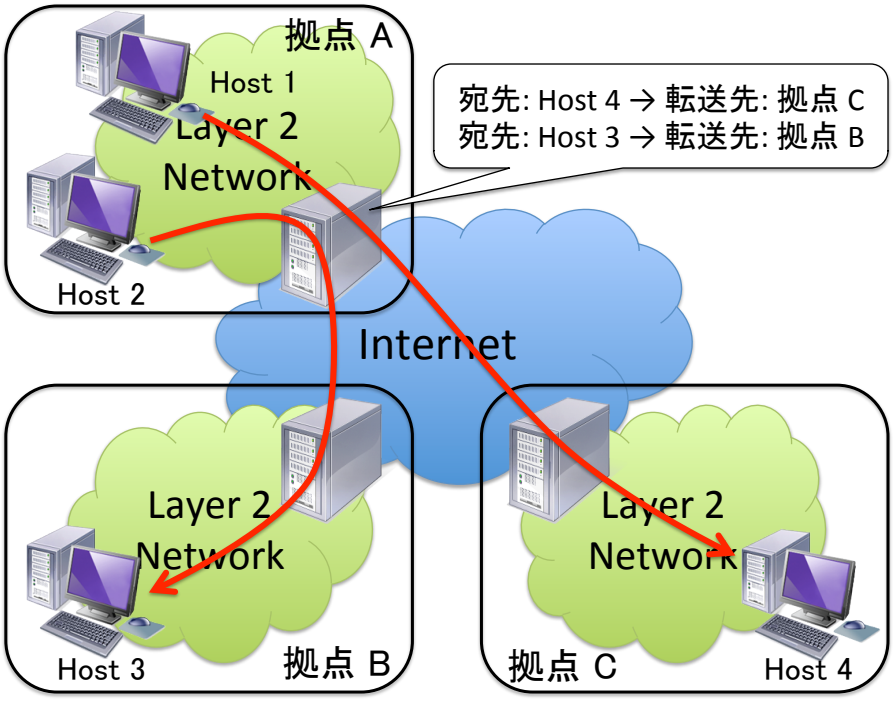

図 2.5: 一対多型の Layer 2 ネットワーク拡張技術

2.2 一対多型の Layer 2 ネットワーク拡張技術

一対多型の Layer 2 ネットワーク拡張技術は複数の拠点に対して同時に Layer 2 ネットワークを拡張することができる。一対一型の Layer 2 ネットワーク拡張技術では、構築された Layer 2 ネットワークが木構造のトポロジーとなってしまうため、必ず一部拠点間の遅延が大きくなってしまうという問題がある。この問題は、Layer 2 ネットワーク拡張技

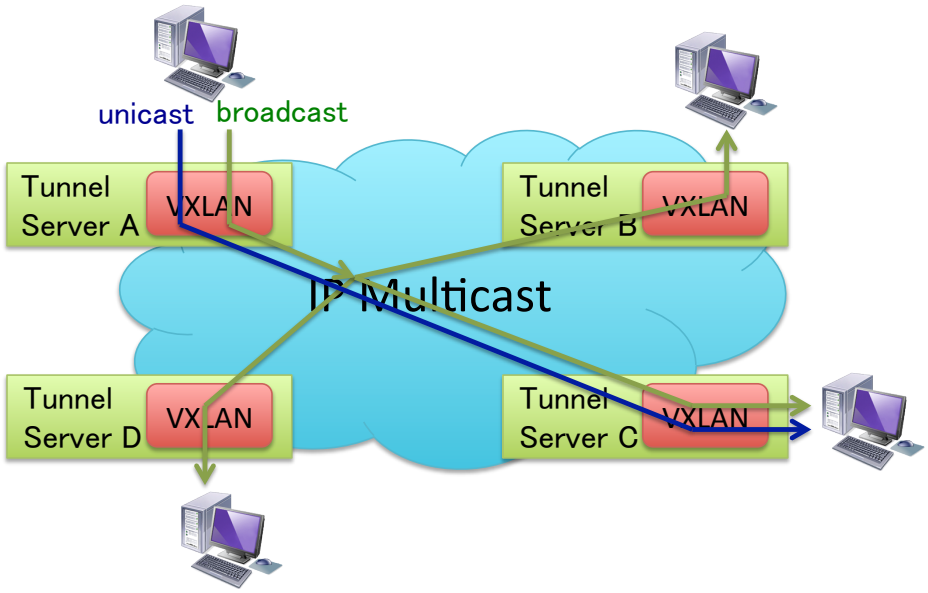

図 2.6: VXLAN を利用して拡張された Layer 2 ネットワーク

術が、図 2.5 で示すように、イーサネットフレームの宛先ホストに応じて転送するトンネル終端点の切り替えを行うことにより解決することができる。Layer 2 ネットワーク拡張技術は、各拠点に設置されているホストを自動的に学習する。そして、イーサネットフレームを転送する際には、宛先ホストがどのトンネル終端点によって収容されているか検索し、そのトンネル終端点にイーサネットフレームを直接転送する。この手法では、一対一型の Layer 2 ネットワーク拡張技術と比べると、イーサネットフレームが宛先のトンネル終端点に直接転送されるため、遅延の小さい Layer 2 ネットワークを構築することができる。

一対多型の Layer 2 ネットワーク拡張技術の例として Virtual Extensible Local Area Network(=VXLAN) [29] と N2N [30] が挙げられる。これらの特徴を以下で述べる。

2.2.1 VXLAN

VXLAN [29] は Cisco Systems 社 [31] や VMware 社 [32] を中心に提案されている一対多型の Layer 2 ネットワーク拡張技術である。VXLAN を利用することにより、同一の Layer 2 ネットワークを複数の拠点に拡張することができる。VXLAN を利用して複数の拠点に同一の Layer 2 ネットワークを拡張した際のトポロジーと VXLAN のイーサネットフレーム転送手法を図 2.6 に示す。VXLAN はトンネル終端点の検知やブロードキャストフレームの転送を行うために、IP マルチキャストを利用する。拡張された Layer 2 ネットワークに参加しているトンネル終端点は、全て同一の IP マルチキャストグループに参加している。トンネル終端点が、収容しているホストからイーサネットフレームを受信すると、まずそのイーサネットフレームの識別を行う。受信したイーサネットフレームが、ブロード

キャストフレームの場合、IP マルチキャストを利用してイーサネットフレームを転送する。全てのトンネル終端点は同一の IP マルチキャストグルームに参加しているため、IP マルチキャストグループに受信したイーサネットフレームを転送することにより、イーサネットフレームは全てのトンネル終端点へ転送される。そして、他のトンネル終端点がそのイーサネットフレームを受信すると、イーサネットフレームを収容しているホストへ転送する。また、イーサネットフレームを受信したトンネル終端点はイーサネットフレームの送信元ホストとイーサネットフレームを転送したトンネル終端点を学習する。受信したイーサネットフレームがユニキャストフレームの場合、まず、イーサネットフレームの宛先ホストが学習されているか確認する。学習されている場合、受信されたイーサネットフレームを宛先ホストが収容されているトンネル終端点に直接転送する。学習されていない場合、ブロードキャストフレームと同様に、IP マルチキャストを利用し全てのトンネル終端点へイーサネットフレームを転送する。そして、宛先ホストが収容されているトンネル終端点がイーサネットフレームを受信すると、そのイーサネットフレームを宛先ホストへ転送する。VXLAN は上記手法により、同時に複数の拠点に同一の Layer 2 ネットワークを拡張する。

VXLAN を利用することにより、一対一型の Layer 2 ネットワーク拡張技術で生じる木構造トポロジーの問題は解決される。VXLAN は転送されてきたイーサネットフレームの送信元ホストと転送したトンネル終端点を自動的に学習する。そして、学習された情報を元に、転送すべきイーサネットフレームの宛先ホストに応じて転送先を自動的に選択する。転送すべきイーサネットフレームがブロードキャストフレームの場合、または、学習されていないホスト宛のユニキャストフレームの場合は IP マルチキャストグループに転送する。学習されているホスト宛のユニキャストフレームの場合は、宛先ホストが収容されているトンネル終端点に直接転送する。これにより、一対一型の Layer 2 ネットワーク拡張技術で生じる不要な中継がなくなるため、遅延は一対一型の Layer 2 ネットワーク拡張技術と比べると小さくなる。

しかし、VXLAN はインターネット上で動くように設計されていないため、VXLAN を利用してインターネット上に分散した複数の拠点に同一の Layer 2 ネットワークを拡張するのは困難である。VXLAN はトンネル終端点の検知と一部イーサネットフレームの転送に IP マルチキャストを用いる。そのため、全てのトンネル終端点は同一の IP マルチキャストグループに参加していることが求められる。しかし、インターネット上に分散した複数の拠点に同一の IP マルチキャストグループを拡張することは困難である。XCast [33] や ScatterCast [34] などといった IP マルチキャストを利用して、インターネット上に分散した複数の拠点に同一の IP マルチキャストグループを拡張することは可能だが、Layer 2 ネットワークを拡張をするために動かさなければいけないシステムが増えるため、運用コストが高くなる上に障害発生時の問題切り分けも難しくなる。そのため、VXLAN はインターネット上の複数の拠点に同一の Layer 2 ネットワークを拡張する手法として適切ではない。

また、VXLAN を利用してインターネット上の複数の拠点に同一の Layer 2 ネットワークを拡張した場合、イーサネットフレームは遅延が最も小さい経路で転送されない。インターネット上の経路には 1.2 節で説明したように、直接転送した場合より、他のトンネル

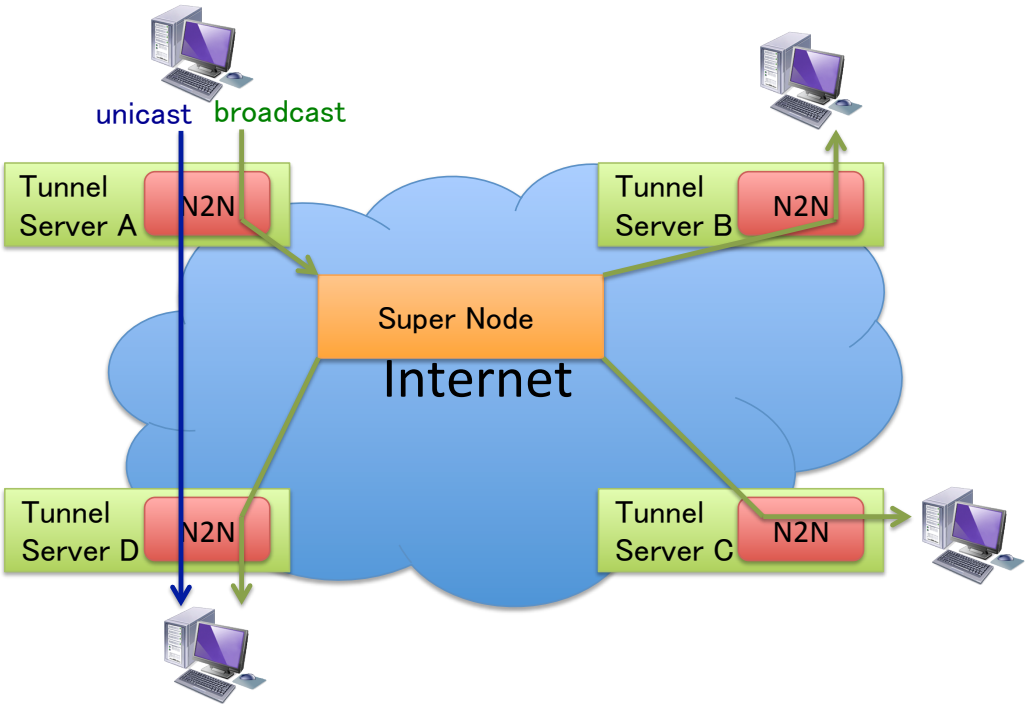

図 2.7: N2N を利用して拡張された Layer 2 ネットワーク

終端点を経由したほうが遅延が小さくなる場合がある。VXLAN はこのような経路が存在した場合でも、それを検知する仕組みが存在しないため、イーサネットフレームの転送は必ず宛先のトンネル終端点に直接行う。そのため、拡張された Layer 2 ネットワークの遅延は最小限とならない。

2.2.2 N2N

N2N [30] は ntop 社 [35] によって開発されている一対多型の Layer 2 ネットワーク拡張技術である。N2N を利用することにより、インターネット上に分散した複数の拠点に同一の Layer 2 ネットワークを拡張することができる。N2N を利用してインターネット上に分散した複数の拠点に同一の Layer 2 ネットワークを拡張した際のトポロジーと N2N のフレーム転送手法を図 2.7 に示す。

N2N はインターネット上に分散した複数の拠点に同一の Layer 2 ネットワークを拡張するために、トンネル終端点とは別に、Supernode というサーバーを用いる。Supernode の役割は大きく分けて 2 つある。1 つ目の役割は、トンネル終端点の参加や離脱などといったコントロールメッセージの通知と管理である。Supernode はトンネル終端点が Layer 2 ネットワークへ参加した際に、Layer 2 ネットワークに参加している全トンネル終端点にそれを通知する。新規トンネル終端点は Supernode の IP アドレスとポート番号を指定するだけで、Layer 2 ネットワークに参加することができる。2 つ目の役割は、ブロードキャストフレームと一部ユニキャストフレームの転送である。トンネル終端点がブロードキャス

トフレームを転送するためには、Supernode にブロードキャストフレームを転送し、Layer 2 ネットワークに参加している全トンネル終端点にブロードキャストフレームを転送してもらう。これにより N2N はインターネット上に分散した複数の拠点に同一の Layer 2 ネットワークを拡張することを可能としている。

N2N はイーサネットフレームを宛先のトンネル終端点に直接転送するため、一対一型の Layer 2 ネットワーク拡張技術のように、一部通信の遅延が大きくなってしまうという問題が生じない。N2N によって拡張された Layer 2 ネットワークに参加しているホストがイーサネットフレームを送信すると、N2N のトンネル終端点はそのイーサネットフレームの転送を行う。トンネル終端点が受信したイーサネットフレームがブロードキャストフレームの場合、トンネル終端点はそのイーサネットフレームを Supernode に転送する。そして、Supernode は受信したイーサネットフレームを全てのトンネル終端点に転送する。Supernode が再転送を行ったブロードキャストフレームをトンネル終端点が受け取ると、収容しているホストにイーサネットフレームを転送した上で、送信元ホストと転送をしたトンネル終端点を学習する。トンネル終端点が受信したイーサネットフレームがユニキャストフレームの場合、ユニキャストフレームの宛先ホストが学習されているホストかを確認する。学習されているホストの場合は宛先ホストが収容されているトンネル終端点にイーサネットフレームを直接転送する。学習されていないホストの場合はブロードキャストフレームと同様に、Supernode へ転送し、全トンネル終端点に転送を行なってもらう。N2N はこのようにイーサネットフレームを宛先のトンネル終端点に直接転送をするため、遅延は一対一型の Layer 2 ネットワーク拡張技術と比べ小さくなる。

しかし、N2N は 1.2 節で説明したようなインターネットの経路が考慮されていない。また、一部のイーサネットフレームは必ず Supernode を経由する。そのため、遅延が最も小さい経路で転送されていないと言える。N2N にはトンネル終端点間の遅延を計測し、その計測結果に基いて遅延が最も小さい経路を選択する仕組みが存在しない。そのため、他のトンネル終端点を経由することにより遅延が小さくなるような経路が存在したとしても、イーサネットフレームは宛先のトンネル終端点に直接転送される。また、ブロードキャストフレームや一部のユニキャストフレームは必ず Supernode を経由する。そのため、トンネル終端点から Supernode までの遅延が大きい場合や、Supernode の負荷が高い場合は遅延が大きくなってしまう。よって、N2N ではインターネット上に分散した複数の拠点に拡張された Layer 2 ネットワークの遅延は最小限とならない。

更に N2N は Supernode が単一障害点となっている。N2N は一部イーサネットフレームの転送に Supernode を用いている。また、Layer 2 ネットワークに参加しているトンネル終端点の管理は Supernode が行なっている。そのため、Supernode で障害が発生した場合、拡張された Layer 2 ネットワークが完全に利用できなくなってしまうという問題がある。

2.3 関連研究の比較

本節では、 2.1 節、 2.2 節で説明した Layer 2 ネットワーク拡張技術について、インターネット上に分散された複数の拠点に同一の Layer 2 ネットワークを拡張した際に、拡張さ

れた Layer 2 ネットワーク上で動作するサービスの遅延によるパフォーマンス低下を小さくするという目的に向けた観点から比較、検討を行う。これを実現するための要件は以下の通りである。

- 宛先に応じた転送先の選択が可能である
- 遅延が最も小さくなる経路の選択が可能である
- インターネット上で動作する
- 分散して動作する

目的を実現するための各要件について、 2.1 節と 2.2 節で挙げた既存研究がどの程度満たしているかを表 2.1 に示す。

表 2.1: 既存研究の比較

既存研究	転送先の選択	遅延に基づいた経路選択	インターネットでの動作	分散
L2TP,GRE	×	×	○	×
VXLAN	○	×	×	○
N2N	○	×	○	×

1.4 節で説明したように、インターネット上に分散された複数の拠点に拡張した Layer 2 ネットワークで動作するサービスの遅延によるパフォーマンス低下を小さくするためには、イーサネットフレームを遅延が最も小さくなる経路で転送する。L2TP や GRE といった一対一型の Layer 2 ネットワーク拡張技術は、転送先をイーサネットフレームの宛先に応じて選択することができない。そのため、Layer 2 ネットワークのトポロジーは木構造となるため、一部通信の遅延は大きくなる。一方、N2N と VXLAN は宛先に応じて転送先を選択することができる。しかし、N2N と VXLAN には、トンネル終端点間の遅延を計測し、その計測結果に基いて遅延が最も小さくなる経路を選択する仕組みがない。そのため、転送する際の経路が、必ずしも遅延の最も小さい経路ではないという問題がある。

また、インターネット上に分散された複数の拠点に同一の Layer 2 ネットワークを拡張するためには、Layer 2 ネットワーク拡張技術がインターネット上で動作する必要がある。L2TP、GRE と N2N はインターネット上で動作するように設計されている。しかし、VXLAN は他トンネル終端点の検知と一部イーサネットフレームの転送に IP マルチキャストを用いているため、インターネット上では動作しないという問題がある。

更に、 1.1.2 項で説明したように、1 つの拠点で発生した障害によるサービスの停止を防ぐために、複数の拠点を利用する場合が多い。拡張された Layer 2 ネットワークが、ある拠点で発生した障害の影響を受け、利用できなくなっては複数拠点の利点を得ることが出来ない。VXLAN は分散して動作しているため、IP マルチキャストが正常に動作していれば、障害の影響は受けない。一方で、L2TP と GRE は木構造のトポロジーとなるた

め、ある拠点で障害が発生すると拡張された Layer 2 ネットワークでも障害が発生する、という問題がある。また、N2N は Supernode を利用して Supernode の管理と一部イーサネットフレームの転送を行なっているため、Supernode が設置されている拠点で障害が発生すると、Layer 2 ネットワークで一切通信ができなくなる、という問題がある。

第3章　提案手法

本章では、低遅延な広域 Layer 2 ネットワークを構築するにあたり想定する環境を整理した上で、本研究で提案するシステムの概要について述べる。

3.1　設計

本研究では、まずトンネル終端点が、トンネル終端点間の遅延を計測し、その計測結果に基づいた経路選択が行えるようにする。例えば、図 3.1 で示すように、藤沢、自宅とソウルの 3 拠点に同一の Layer 2 ネットワークを拡張するとする。この場合、藤沢とソウルの間で通信するにあたり、インターネットの経路で直接転送するよりも、自宅を経由することにより約 35%小さい遅延で通信することが可能である。Layer 2 ネットワーク拡張技術は、このような経路を発見するために、Layer 2 ネットワークに参加している全トンネル終端点の遅延を計測する。そして、遅延が最も小さくなる経路を計測結果から計算し、その経路でイーサネットフレームの転送を行う。

また、インターネット上で分散して動作するよう、トンネル終端点が個々で Layer 2 ネットワークに参加しているトンネル終端点のリストを管理するようにする。トンネル終端点は Layer 2 ネットワークへの参加や離脱などのメッセージを、Layer 2 ネットワークに参加している全てのトンネル終端点に広告する。これにより、全てのトンネル終端点で Layer 2 ネットワークに参加しているトンネル終端点のリストを共有する。また、全てのトンネル終端点に転送する必要があるイーサネットフレームは、Supernode や IP マルチキャストに転送するのではなく、全てのトンネル終端点に 1 つ 1 つ転送する。これによ

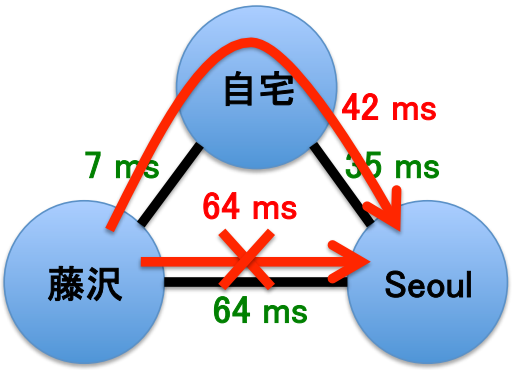

図 3.1: 他拠点を経由させることによる遅延の削減

り、Supernode や IP マルチキャストが不要となり、インターネット上で分散して動作する Layer 2 ネットワークを構築することができる。

3.2 想定環境

本研究の目的は、インターネット上でサービスを提供しているサービスプロバイダーが、インターネット上に分散された複数の拠点に同一の Layer 2 ネットワークを拡張する際に、拡張された Layer 2 ネットワーク上で動作するサービスの遅延によるパフォーマンス低下を小さくすることである。拠点の数は数十拠点を想定する。また、拠点の場所は国内、及び、その近隣諸国とする。インターネット上に拠点が分散しているような環境では、現在のインターネットに存在する経路の問題により、宛先と直接通信するより、別の拠点を経由して宛先に到達するほうが低遅延で通信できる場合がある。そこで本研究では、遅延を小さくするために、イーサネットフレームを転送する際に遅延の最も小さい経路で転送をすることができる Layer 2 ネットワーク拡張技術を提案する。これが実現することにより、従来の手法と比べ、拡張された Layer 2 ネットワーク上で動作するアプリケーションのパフォーマンスが向上されることが期待される。

本研究では拡張された Layer 2 ネットワーク上で利用されるアプリケーションとして NFSv3 を想定する。複数の拠点に分散したサービスを構築するためには、同じデータを全拠点からアクセスできることが必要となる場合がある。複数の拠点から共通のデータを利用する手段として NFSv3 を利用するという手法が挙げられる。しかし、NFSv3 は遅延がとても小さい、単一の拠点内で運用されるように設計されているため、遅延の大きい環境で利用した場合、パフォーマンスが著しく低下する。本研究ではこのような低遅延を要求するアプリケーションを想定アプリケーションとする。

3.3 機能要件

前節で説明したような環境で、低遅延な拡張された Layer 2 ネットワークを実現するための Layer 2 ネットワーク拡張技術に対する要件は以下の通りである。

- インターネット上に分散された複数の拠点への Layer 2 ネットワークの拡張
- 宛先に応じた転送先の選択
- 遅延の最も小さい経路でのイーサネットフレームの転送
- 分散して動作すること

低遅延な拡張された Layer 2 ネットワークを構築するためには、遅延の最も小さい経路でイーサネットフレームが転送される必要がある。これを実現するために、まず Layer 2 ネットワーク拡張技術は参加している全てのトンネル終端点間の遅延を計測し、その計測結果を用いて遅延が最も小さい経路を計算する。そしてトンネル終端点が、イーサネット

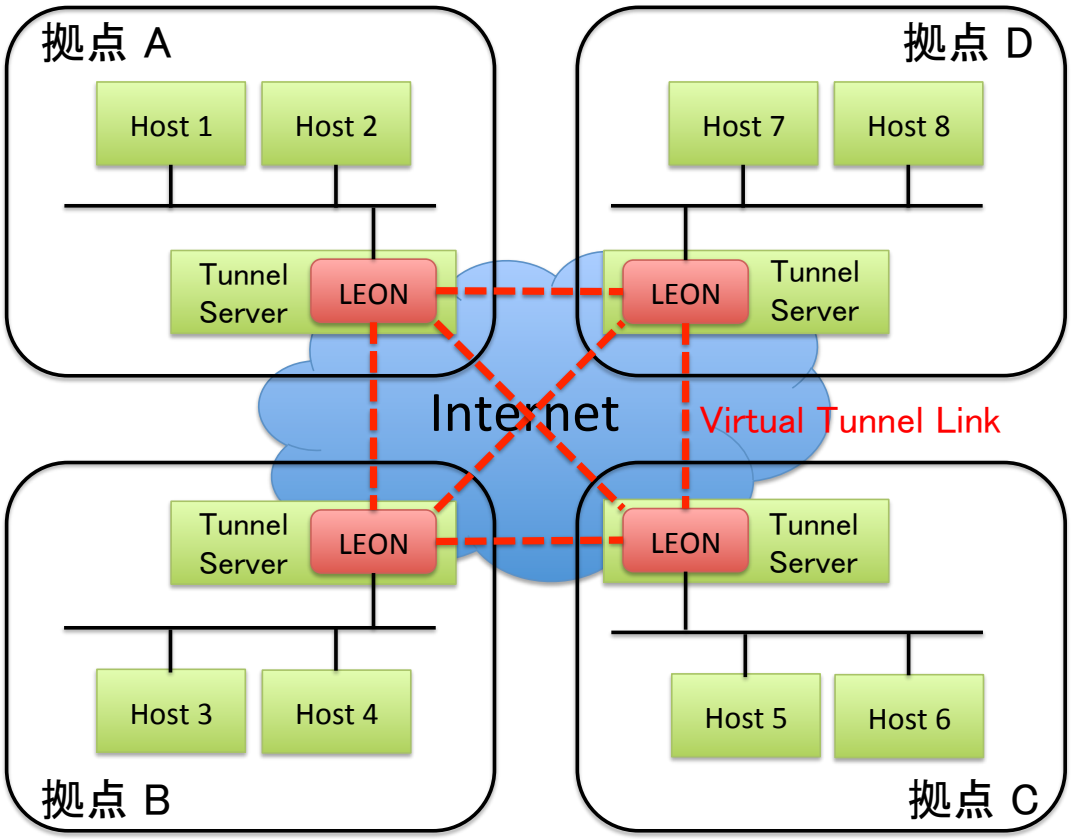

図 3.2: LEON によって構築される Layer 2 ネットワークトポロジー

フレームを転送する際には、イーサネットフレームの宛先に応じて転送するトンネル終端点を選択し、計算した経路に基いて転送を行う。宛先のトンネル終端点に直接転送する場合が最も遅延の小さい経路の場合には直接転送をする。他のトンネル終端点を経由して宛先に転送したほうが小さい遅延で転送できる場合は、他のトンネル終端点を中継する。また、インターネット上に分散された複数の拠点に Layer 2 ネットワークを拡張することができる必要がある。さらに、どこかの拠点で障害が発生した場合でも、正常に Layer 2 ネットワークが動作していることが求められる。そのため、Supernode などといった中心となるサーバーや IP マルチキャストが必要なく、分散して動作する必要がある。

3.4 システム概要

本研究では、前節で示した機能要件を実現する低遅延 Layer 2 ネットワーク拡張手法として、Latency Efficient Overlay Network (LEON) を提案する。LEON は IP ネットワークを経由してイーサネットフレームを転送する Layer 2 ネットワーク拡張技術である。LEON によって構築される Layer 2 ネットワークのトポロジーを図 3.3 に示した。LEON を動かし、トンネル終端点として動作するサーバーは各拠点に 1 つ設置する。トンネル終端点として動作するサーバーはインターネットと拡張された Layer 2 ネットワークに所属するホ

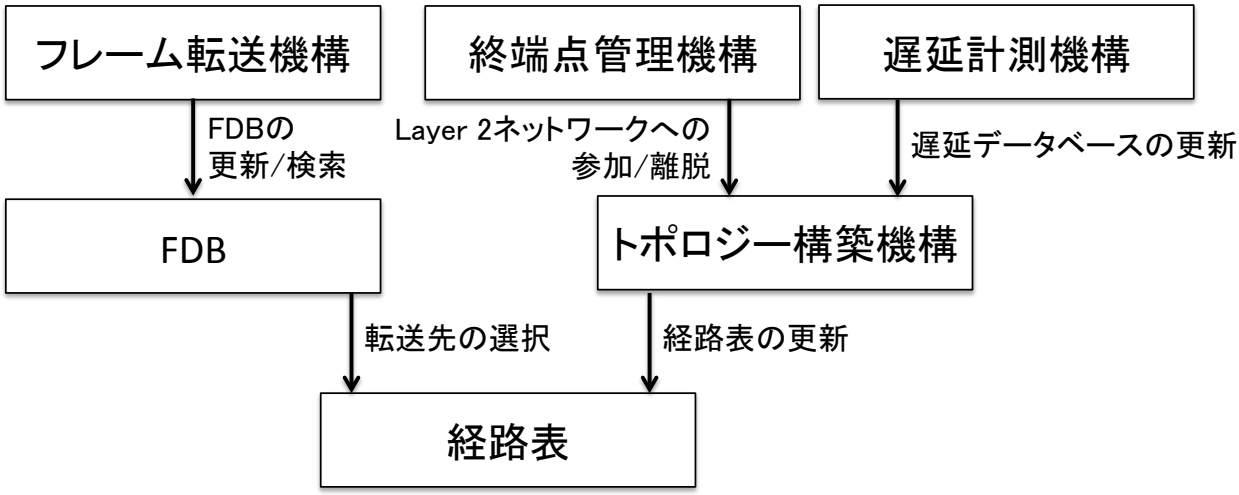

図 3.3: LEON のシステム概要

ストが収容されている Layer 2 ネットワークに接続されている。そして、LEON は自分の拠点に所属しているホストが、他拠点に設置されているホストと通信するためのゲートウェイとして動作する。

LEON のシステム概要を図 3.3 に示す。まず、LEON は既に Layer 2 ネットワークに参加しているトンネル終端点を学習する。トンネル終端点の学習後、トンネル終端点間の遅延を計測する。トンネル終端点間の遅延は時間帯に応じて変化する可能性があるため、トンネル終端点の学習時のみだけでなく、定期的に実行される。また、遅延の計測結果は他トンネル終端点に広告する。そして、全てのトンネル終端点への遅延計測が完了後、各トンネル終端点と通信する際に遅延が最も小さくなる経路を計算する。ここで計算された経路は経路表にする。LEON が転送すべきイーサネットフレームを受信した際には、ホストがどのトンネル終端点に収容されているかが記録されているデータベースを参照し、イーサネットフレームの宛先ホストがどのトンネル終端点によって収容されているのか検索する。そして、事前に計算された経路表に基づき、イーサネットフレームを宛先ホストが収容されているトンネル終端点へ転送する。

トンネル終端点の学習は、トンネル終端点を管理するサーバーなどを用いず、トンネル終端点間で自律的に行う。トンネル終端点を管理するサーバーを用意することにより、Layer 2 ネットワークに参加しているトンネル終端点の学習やトンネル終端点間のメッセージングは容易となる。しかし、トンネル終端点を管理するサーバーが単一障害点となってしまい、障害発生時には拡張された Layer 2 ネットワークの通信全体に影響を与えてしまう可能性がある。トンネル終端点が自律的に他のトンネル終端点を学習できた場合、トンネル終端点を管理するサーバーが必要なくなり、どれかのトンネル終端点で障害が発生したとしても、影響を受けるのは障害が発生しているトンネル終端点だけである。そのため、LEON では他トンネル終端点をトンネル終端点が自律的に学習できるようにする。

学習されたトンネル終端点へ到達するための経路は、トンネル終端点間の遅延に基いて選択する。 1.2 節で説明したように、現在のインターネットの経路は宛先までの遅延やリ

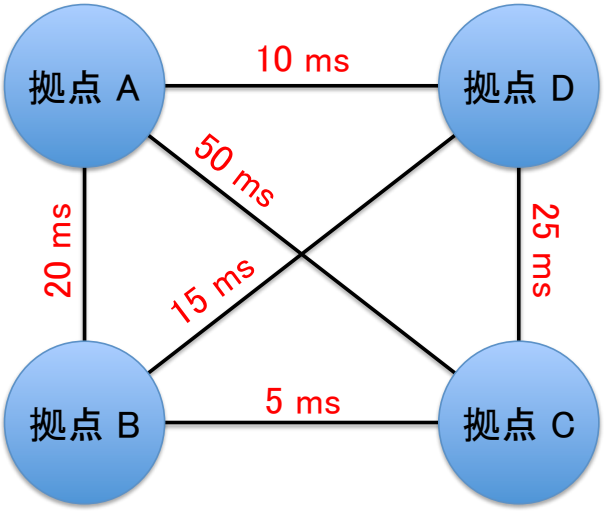

図 3.4: RTT をコストとした経路の選択

ンクの使用率などが考慮されていないため、遅延が最も小さい経路でない場合がある。宛先にインターネットの経路で到達するよりも、他のトンネル終端点を経由させたほうが遅延が小さくなる場合がある。そのため、遅延が最も小さい経路でイーサネットフレームを転送するためには、トンネル終端点は宛先のトンネル終端点に転送する際に、遅延が最も小さい経路を計算する必要がある。LEON は、遅延が最も小さくなる経路を計算するために、トンネル終端点間の遅延を計測する。また、計測された遅延の計測結果は、他のトンネル終端点が同様に計算するために必要となるため、他のトンネル終端点にも広告する。収集された遅延の計測結果は、図 3.4 のようなトンネル終端点間の遅延関係を計算するために利用される。そして、この遅延関係から遅延をコストとしたダイクストラ法 [36] を用いて、Layer 2 ネットワークに参加している各トンネル終端点に到達するにあたり遅延が最も小さくなる経路を計算する。宛先のトンネル終端点に到達するにあたり、インターネットの経路に基いて、直接転送する経路が遅延が最も小さい経路であれば直接転送を行う。宛先のトンネル終端点に到達するにあたり、他のトンネル終端点を経由する経路が遅延の最も小さい経路であれば、そのトンネル終端点に中継してもらう。LEON では、このような手法を用いて、Layer 2 ネットワークに参加している各トンネル終端点へ到達するための経路を選択する。

また、イーサネットフレームの転送は、前述した各トンネル終端点までの経路とは別に、どのホストがどのトンネル終端点によって収容されているかを記録している Fowarding Database(=FDB) に従って行われる。トンネル終端点は、イーサネットフレームを転送するにあたり、宛先ホストがどのトンネル終端点によって収容されているのかという情報を事前に記録されている必要がある。LEON では、ホストから送られてくるブロードキャストフレームから、自動的に FDB の更新を行う。他トンネル終端点からブロードキャストフレームが送られてきた際には、ブロードキャストフレームの送信元 MAC アドレスとそのイーサネットフレームを転送したトンネル終端点を FDB に記録する。既知の送信元

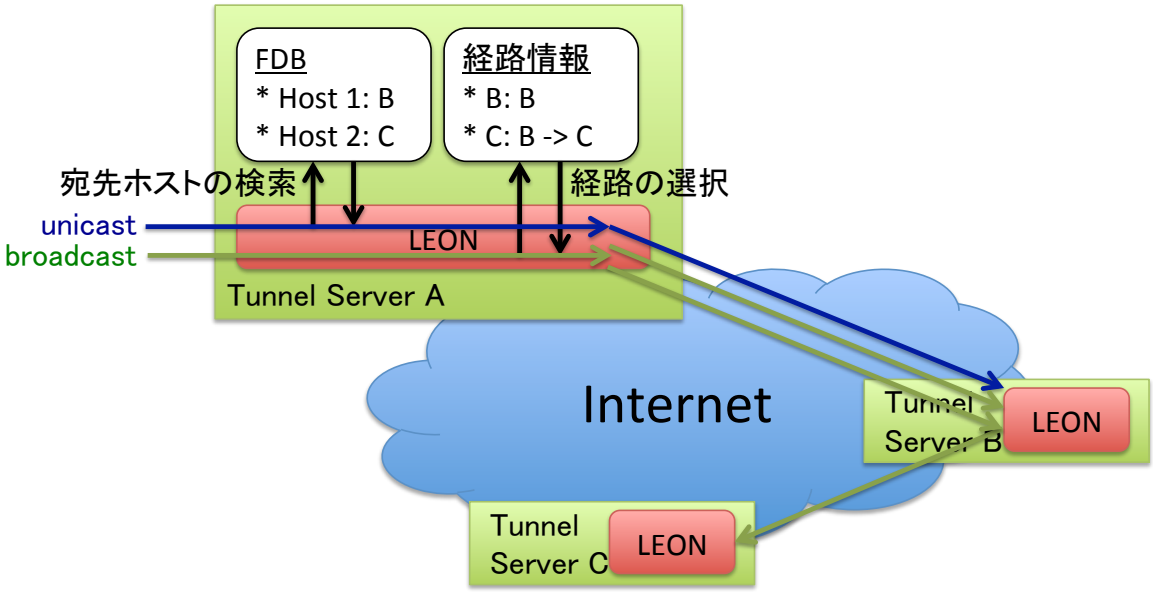

図 3.5: 転送動作の概要

MAC アドレスが、記録されているトンネル終端点とは違うトンネル終端点から転送されてきた際には、FDB の更新を行う。そして、イーサネットフレームを送信する際には、宛先ホストがどのトンネル終端点によって収容されているのかを FDB から検索し、その情報と事前に計算を行った経路に基いて転送を行う。

イーサネットフレームの転送は、各トンネル終端点がそれぞれ持つ経路表と自動的に学習された FDB をもとに、宛先ホストまでの遅延が最も小さい経路で転送される。転送動作の概要を図 3.5 に示す。トンネル終端点が転送するイーサネットフレームを受け取ると、そのイーサネットフレームがユニキャストフレームかブロードキャストフレームかを判別する。ユニキャストフレームの場合、宛先ホストがどのトンネル終端点によって収容されているか FDB から検索する。そして、経路表に従い、宛先までの遅延が最も小さい経路で転送するための転送先を決定する。ブロードキャストフレームの場合は、宛先ホストが所属しているトンネル終端点として全てのトンネル終端点を選択し、それぞれのトンネル終端点に転送するにあたり、遅延が最も小さい経路で転送するための転送先を決定する。インターネットの経路で宛先のトンネル終端点に到達することにより、遅延が最も小さく転送できるのであれば、転送先は宛先のトンネル終端点となる。他のトンネル終端点を中継させることにより、遅延がインターネットの経路で直接転送した時よりも小さくなるのであれば、転送先は中継するトンネル終端点となる。転送先が決定されると、イーサネットフレームに転送経路が書かれたヘッダーと転送先のトンネル終端点が宛先となっている IP ヘッダーが追加され、インターネットにエンキャプシュレーションされたイーサネットフレームが転送される。ユニキャストフレームの場合、転送先が 1 つなのでエンキャプシュレーション作業は 1 回のみである。ブロードキャストフレームの場合、転送先が複数あるので、Layer 2 ネットワークに参加しているトンネル終端点の台数と同じ回数、エンキャプシュレーション作業が行われる。LEON ではこのように、遅延が最も小さい経

路で、イーサネットフレームの転送を行う。

トンネル終端点が、エンキャプシュレーションされたイーサネットフレームを受信した際には、イーサネットフレームに追加された LEON ヘッダーに記述された経路に応じて転送を行う。Layer 2 ネットワーク上のホストから、他のトンネル終端点によって収容されているホストが宛先であるイーサネットフレームを受信したトンネル終端点は、そのイーサネットフレームを転送する際に経路表から転送する経路を選択する。選択された経路は、イーサネットフレームを転送する際に、イーサネットフレームの先頭に追加される LEON ヘッダーに記述される。そして、トンネル終端点はエンキャプシュレーションされたイーサネットフレームを受け取ると、LEON ヘッダーに記述された経路を確認する。そのトンネル終端点が記述された経路の最後に書かれたトンネル終端点の場合は、そのトンネル終端点宛のイーサネットフレームであると判断し、送信元トンネル終端点によって追加された IP ヘッダーと LEON ヘッダーを取り外し、Layer 2 ネットワーク上の宛先ホストへ転送をする。そのトンネル終端点が経路表の最後に書かれたトンネル終端点でない場合は、そのトンネル終端点の次に記述されているトンネル終端点へイーサネットフレームをさらに転送する。これにより、トンネル終端点は受信したエンキャプシュレーションされたイーサネットフレームの転送先を決定する。

これにより、LEON は遅延が最も小さい経路でイーサネットフレームが転送される Layer 2 ネットワークの構築を行う。LEON は Layer 2 ネットワークに参加しているトンネル終端点間の遅延を計測し、計測結果に基づき、イーサネットフレームの転送先を選択する。最も遅延が小さくなる経路が、インターネット上の経路に基づいて転送する経路の場合、インターネット上の経路でイーサネットフレームが転送される。他のトンネル終端点を経由する経路が遅延の最も小さい経路の場合、経由すべきトンネル終端点に転送される。転送する際に、LEON は転送経路が記述された LEON ヘッダーをイーサネットフレームに追加する。そして、転送されてきたイーサネットフレームを受信した際には、LEON ヘッダーのイーサネットフレームを参照し、受信したトンネル終端点が経路の最後に記述されたトンネル終端点の場合は、イーサネットフレームの宛先ホストへイーサネットフレームを転送する。受信したトンネル終端点が経路の最後に記述されたトンネル終端点でない場合は、次に記述されているトンネル終端点へ受信したイーサネットフレームを転送する。これにより、LEON はイーサネットフレームをインターネット上で遅延が最も小さい経路で転送する。

3.5 プロトコルの設計

LEON は、遅延が最も小さい経路でイーサネットフレームが転送される Layer 2 ネットワークの構築と利用、そしてトンネル終端点間の遅延を計測するために大きく分けて 3 種類のプロトコルを利用する。まず 1 つ目として、Layer 2 ネットワークの構築を行う際に利用されるコントロールプロトコルである。トンネル終端点は、コントロールプロトコルを利用して、Layer 2 ネットワークへの参加と離脱を行う。2 つ目として、イーサネットフレームの転送を行う際に利用される転送プロトコルが挙げられる。転送プロトコルを

利用して、Layer 2 ネットワーク上のイーサネットフレームがトンネル終端点間でやり取りされる。そして、3 つ目として、トンネル終端点間の遅延情報の収集を行う際に利用される計測プロトコルが挙げられる。計測プロトコルを利用して、トンネル終端点間の遅延の計測とその計測結果のトンネル終端点間での共有を行う。これらのプロトコルは、全て UDP を用いる。LEON が用いるメッセージの共通ヘッダーフォーマットを図 3.6 に示す。

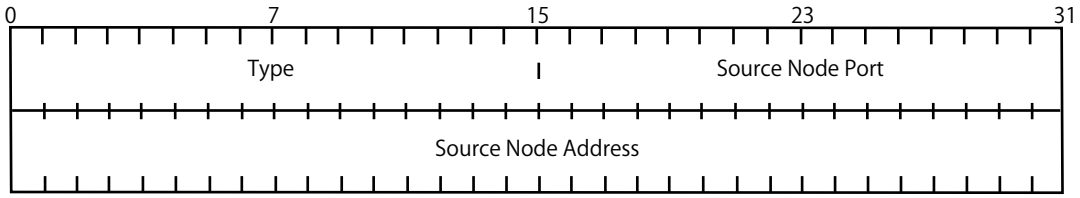

図 3.6: LEON の共通ヘッダーフォーマット

Type フィールドは他トンネル終端点から送られてきたメッセージを識別するために利用される。Type の種類は全てで 8 種類である。まず、トンネル終端点が Layer 2 ネットワークへ参加と離脱を行うためのコントロールプロトコルで利用される Type として JOINNODE(20)、JOINNODEACK(21)、DELNODE(30)、REQALL(40) の 5 種類がある。次に、トンネル終端点間でイーサネットフレームの転送を行うための転送プロトコルで利用される Type として FORWARD(10) の 1 種類がある。最後に、トンネル終端点間の遅延情報の収集を行うための遅延計測プロトコルで利用される Type として PINGREQ(50)、PINGACK(51)、LATENCYDATA(60) の 3 種類がある。Source Node Address と Source Node Port フィールドはメッセージの送信元トンネル終端点の IP アドレスとポート番号が挿入されている。以下に、各プロトコルの詳細について述べる。

3.5.1 コントロールプロトコル

コントロールプロトコルはトンネル終端点が Layer 2 ネットワークに参加と離脱を行う際に用いられる。コントロールメッセージのフォーマットを図 3.7 に示す。

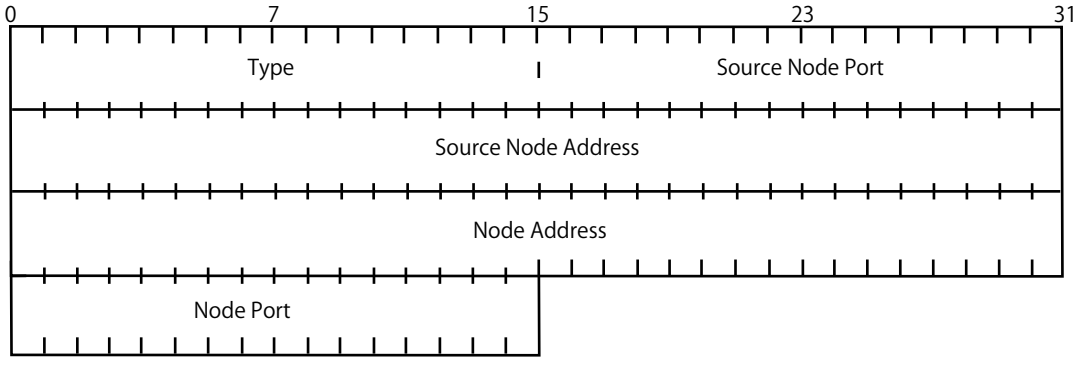

図 3.7: コントロールメッセージのフォーマット

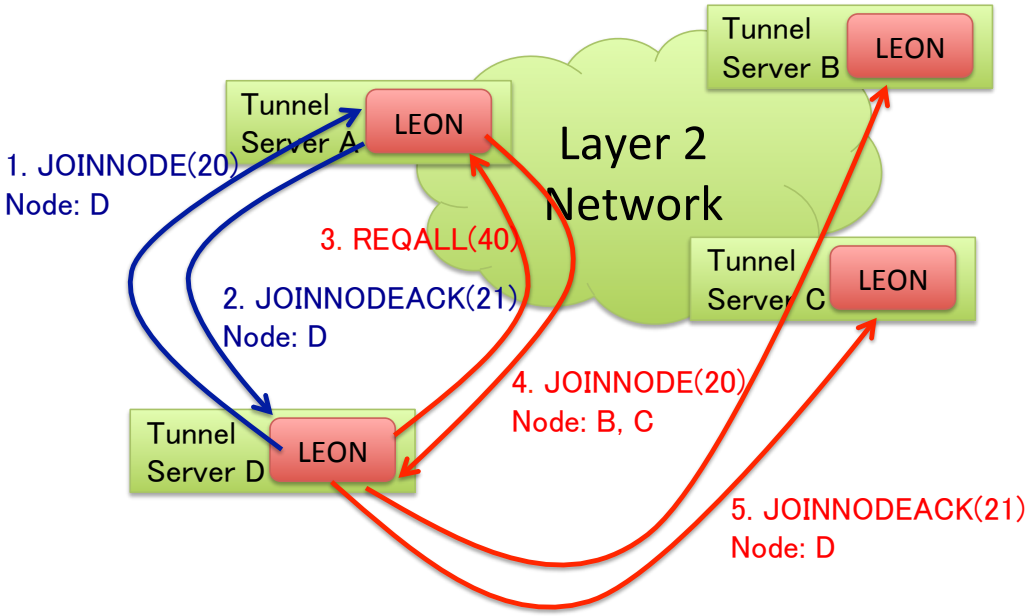

図 3.8: 新規トンネル終端点の参加プロセス

トンネル終端点が新たに既存の Layer 2 ネットワークに参加する際の手順を図 3.8 に示す。まず、既に Layer 2 ネットワークに参加しているトンネル終端点の 1 つに Type フィールドが JOINNODE(20) であるコントロールメッセージを送信する。このときコントロールメッセージの Node Address フィールドと Node Port フィールドには、新たに参加するトンネル終端点の IP アドレスとポート番号が挿入されている。送信されたコントロールメッセージを受信したトンネル終端点は、そのトンネル終端点が管理しているトンネル終端点一覧に送信元トンネル終端点を追加する。そして、正常に受け取ったことを送信元トンネル終端点に通知するために、JOINNODEACK(21) となっているコントロールメッセージを送信する。このときコントロールメッセージの Node Address フィールドと Node Port フィールドは新たに参加したトンネル終端点の IP アドレスとポート番号が挿入されている。新たに参加するトンネル終端点が、JOINNODEACK(21) となっているコントロールメッセージを受信すると、新たに参加するトンネル終端点が管理するトンネル終端点一覧に既に Layer 2 ネットワークに参加しているトンネル終端点の 1 つが登録される。次に、新たに参加するトンネル終端点は、他の既に Layer 2 ネットワークに参加しているトンネル終端点に対しても同様な登録作業を行う必要があるため、登録作業を済ませたトンネル終端点から既に参加している全トンネル終端点の一覧を取得する。一覧を取得するためには、Type フィールドが REQALL(40) であるコントロールメッセージを送信する。このときコントロールメッセージの Node Address フィールドと Node Port フィールドは 0 である。このコントロールメッセージを受信したトンネル終端点は、Type フィールドが JOINNODE(20) であるコントロールメッセージを全トンネル終端点分送り返す。Node Address フィールドと Node Port フィールドに既に参加している各トンネル終端点を挿入し、コントロールメッセージを送信する。新たに参加するトンネル終端点が

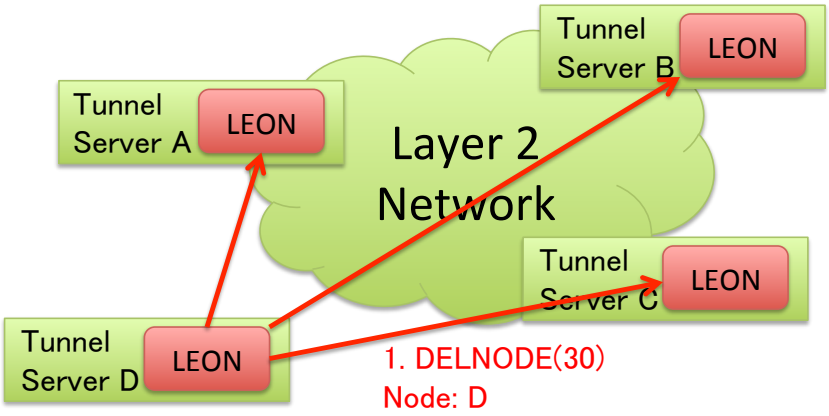

図 3.9: トンネル終端点の離脱プロセス

このコントロールメッセージを受信すると、自身が管理するトンネル終端点一覧に Node Address フィールドと Node Port フィールドに書かれたトンネル終端点を登録する。そして、登録したトンネル終端点に対して、Type フィールドが JOINNODEACK(21) であるコントロールメッセージを送信する。これにより、相手のトンネル終端点に相手が管理しているトンネル終端点一覧に、新たなトンネル終端点の登録をしてもらう。この手順で新たなトンネル終端点は既存の Layer 2 ネットワークに参加する。

トンネル終端点が参加している Layer 2 ネットワークを離脱する際の手順を図 3.9 に示す。離脱するトンネル終端点は、トンネル終端点一覧に入っている全てのトンネル終端点に対して Type フィールドが DELNODE(30) であるコントロールメッセージを送信する。このとき、Node Address フィールドと Node Port フィールドには離脱するトンネル終端点の IP アドレスとポート番号が挿入される。そして、このコントロールメッセージを受信したトンネル終端点は、Node Address フィールドと Node Port フィールドに挿入されたトンネル終端点をトンネル終端点一覧から削除する。また、同時に、そのトンネル終端点によって収容されていたホストを FDB から削除する。これにより、トンネル終端点は参加している Layer 2 ネットワークから離脱される。

3.5.2 転送プロトコル

転送プロトコルはトンネル終端点が受信したイーサネットフレームを転送する際と、トンネル終端点がイーサネットフレームを中継する際に用いられる。転送メッセージのフォーマットを図 3.10 に示す。

転送メッセージは LEON ヘッダーとホストから受信したイーサネットフレームによって構成される。LEON ヘッダーは共通ヘッダーと転送経路情報で成り立っている。転送経路情報は転送メッセージを中継するトンネル終端点と最終的に受信するトンネル終端点のリストである。トンネル終端点は転送メッセージを LEON ヘッダーの転送経路情報に従って転送する。転送メッセージの Forward Type フィールドは、次に書かれたトンネル

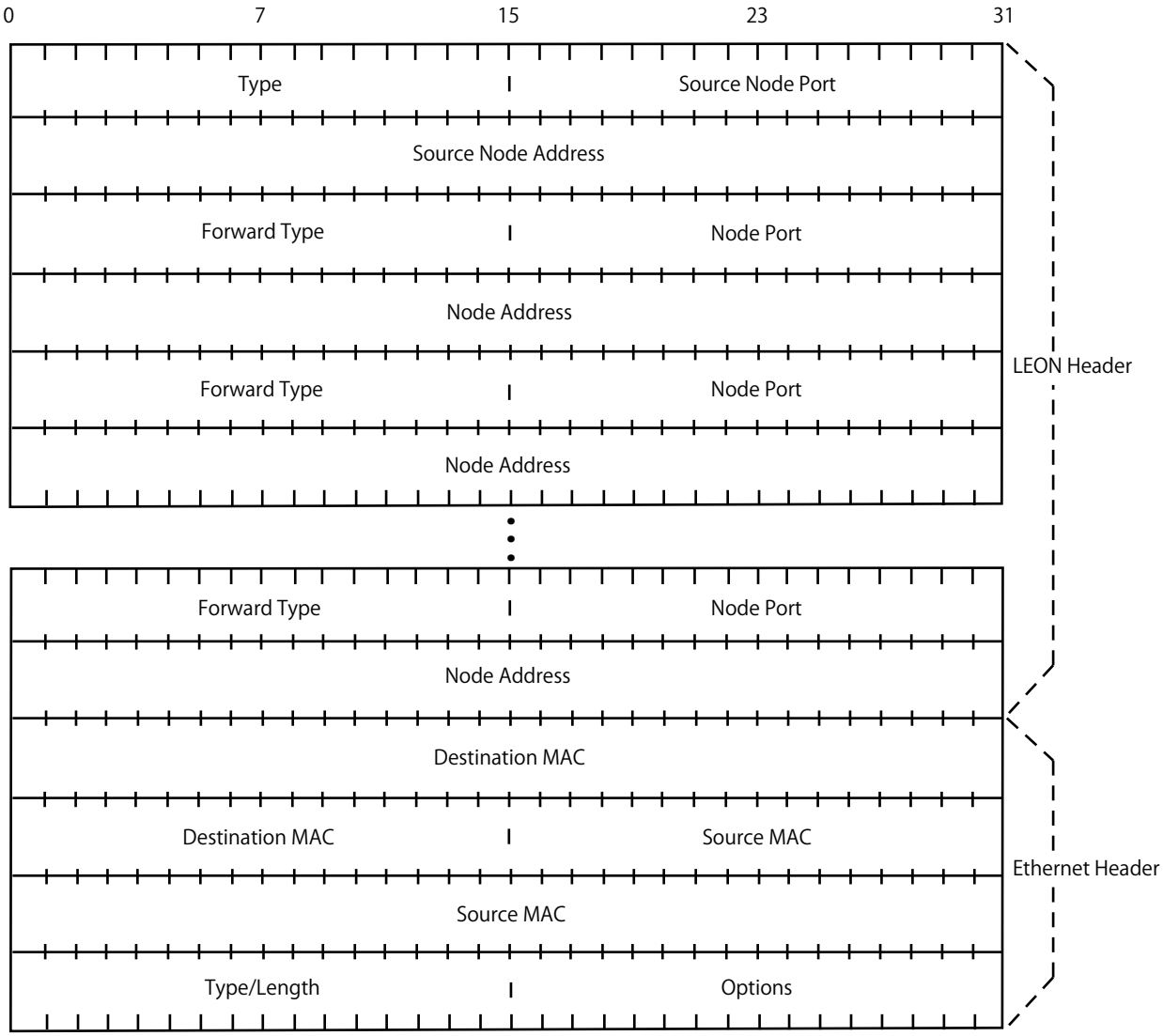

図 3.10: 転送メッセージのパケットフォーマット

終端点が中継するためのトンネル終端点か受信するトンネル終端点かを識別するためのものである。Foward Type フィールドには FORWARD(1) または RECEIVE(0) のどちらかを挿入する。そして、その次の Node Address 及び Node Port フィールドには中継、または、受信をするトンネル終端点の IP アドレスとポート番号を挿入する。転送経路情報の後ろにはホストから受信したイーサネットフレームを付加する。

トンネル終端点が受信したイーサネットフレームを転送する際の手順を図 3.11 に示す。トンネル終端点が収容しているホストからイーサネットフレームを受信すると、イーサネットフレームの Destination MAC フィールドを参照し、宛先のホストがどのトンネル終端点によって収容されているか FDB から検索し、そのトンネル終端点までの経路を選択する。そして、イーサネットフレームの先頭に LEON ヘッダーを追加する。この際の共通ヘッダーの Type フィールドは FORWARD(10)、Source Node Address フィールドと

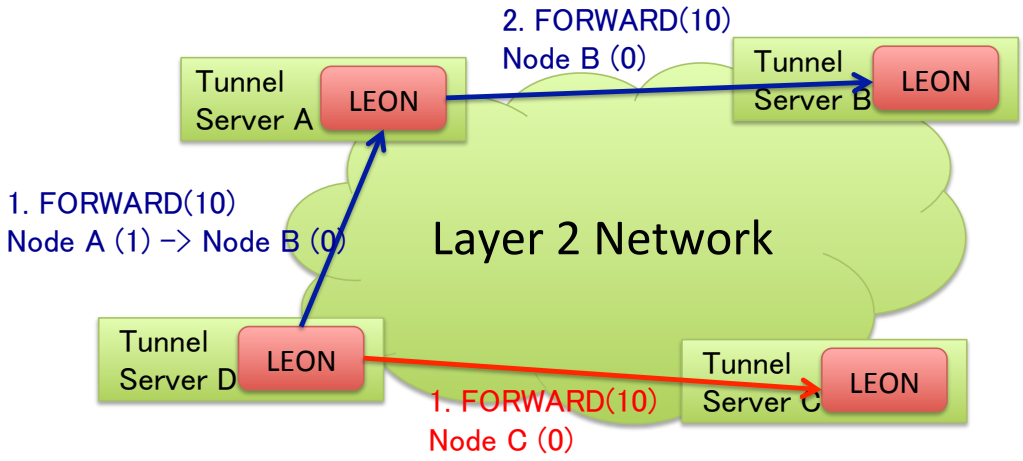

図 3.11: イーサネットフレームの転送プロセス

Source Node Port フィールドはイーサネットフレームを受信したトンネル終端点の IP アドレスとポート番号である。また、転送経路情報の部分には選択された経路が挿入される。イーサネットフレームを直接宛先のトンネル終端点に転送する場合は、転送経路情報は Forward Type フィールドが RECEIVE(0)、Node Address フィールドと Node Port フィールドが宛先トンネル終端点の IP アドレスとポート番号の 1 つである。イーサネットフレームを他のトンネル終端点を中継して転送する場合は、転送経路情報は Forward Type フィールドが FORWARD(1)、Node Address フィールドと Node Port フィールドあ中継するトンネル終端点の IP アドレスとポート番号のヘッダー部分が中継するトンネル終端点分追加される。そして、その後に最終的に受信するトンネル終端点のヘッダー部分を追加する。このとき追加されるヘッダー部分は直接転送する場合のヘッダー部分と同じである。最後に、イーサネットフレームが LEON ヘッダーの後ろに追加される。

トンネル終端点が転送メッセージを受信すると自分を転送経路情報から探し、Forward Type フィールドを確認する。Forward Type フィールドが RECEIVE(0) の場合、受信したトンネル終端点が最終的な宛先であることがわかる。トンネル終端点は LEON ヘッダーを受信したパケットから取り除き、イーサネットフレームを Layer 2 ネットワーク上の宛先ホストへ転送する。Foward Type フィールドが FORWARD(1) の場合、受信したトンネル終端点は中継用のトンネル終端点であることがわかる。トンネル終端点は LEON ヘッダーの転送経路情報から、そのトンネル終端点を転送経路情報からを消去し、次に指定されているトンネル終端点へ転送する。この際、最終的な宛先であるトンネル終端点が、それがどのトンネル終端点から送信されたものか判別できるよう、共通ヘッダーの Source Node Address フィールドと Source Node Port フィールドの変更は行わない。

3.5.3 遅延計測プロトコル

遅延計測プロトコルはトンネル終端点間の遅延の計測と、その計測結果の共有に用いられる。遅延の計測を行うための遅延計測メッセージのフォーマットを図 3.12 に示す。また、計測結果を共有するための遅延データメッセージのフォーマットを図 3.13 に示す。

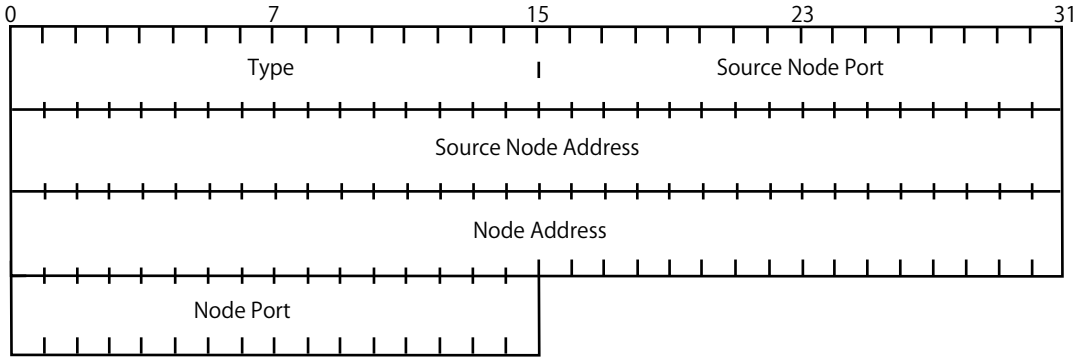

図 3.12: 遅延計測メッセージのフォーマット

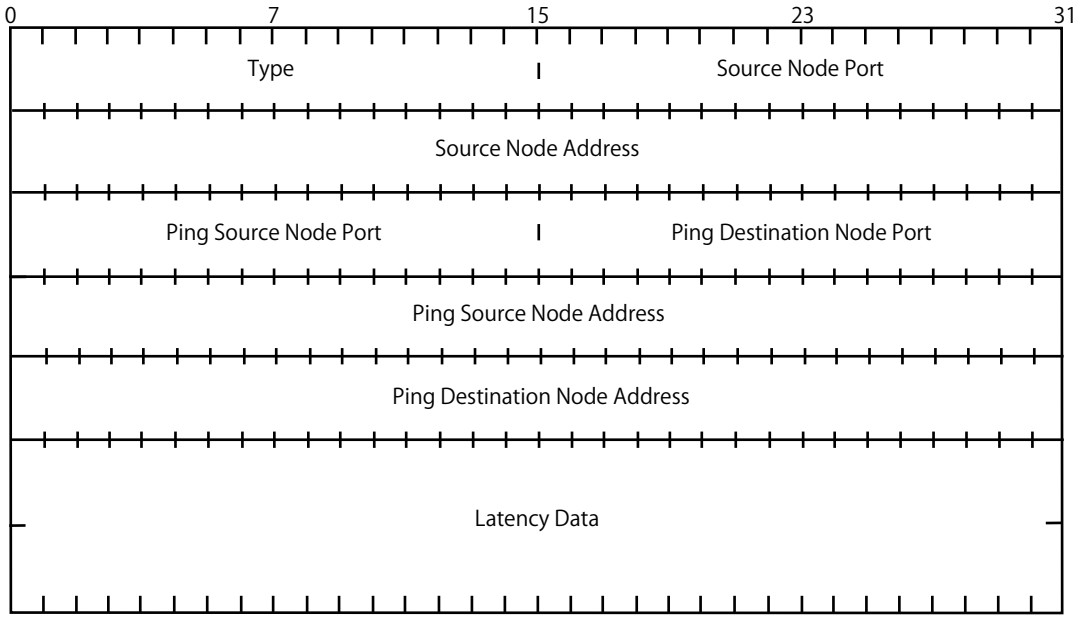

図 3.13: 遅延データメッセージのフォーマット

トンネル終端点がトンネル終端点間の遅延を計測し、その計測結果を共有する際の手順を図 3.14 に示す。トンネル終端点間の遅延計測は定期的に全トンネル終端点で行われる。まず、トンネル終端点はトンネル終端点が管理するトンネル終端点一覧に登録されている全トンネル終端点までの遅延を計測する。遅延の計測を行うには、トンネル終端点は宛先のトンネル終端点に遅延計測メッセージを送信する。遅延計測メッセージの Type

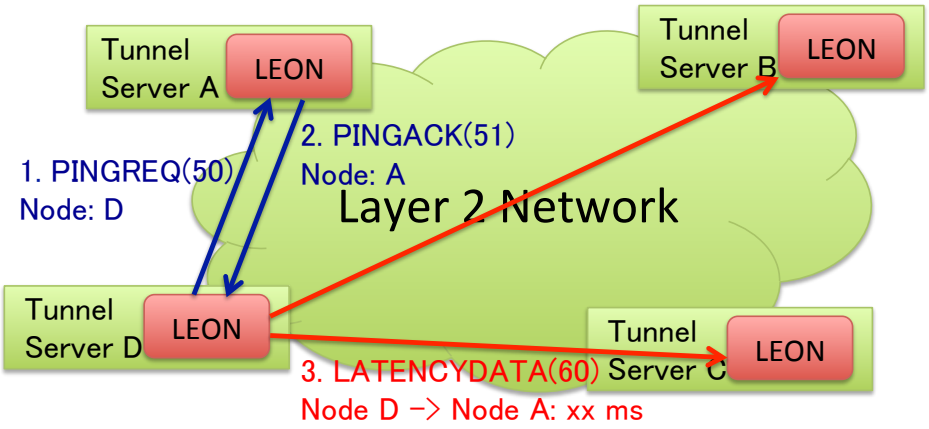

図 3.14: 遅延データの共有プロセス

フィールドには PINGREQ(50)、Source Node Address フィールド、Node Address フィールド Source Node Port フィールドと Node Port フィールドには遅延計測を行うトンネル終端点の IP アドレスとポート番号を挿入する。また、遅延計測を行うトンネル終端点は遅延計測メッセージの送信した時間を記憶する。そして、遅延計測メッセージを受信したトンネル終端点は同様に遅延計測メッセージを返す。この時の遅延計測メッセージの Type フィールドには PINGACK(51)、Source Node Address フィールド、Node Address フィールド、Source Node Port フィールドと Node Port フィールドには返信をするトンネル終端点の IP アドレスとポート番号を挿入する。遅延計測を行うトンネル終端点が返信の遅延計測メッセージを受け取ると、受け取った時間から送信した時間の差から遅延を求める。そして、遅延を遅延データベースに登録する。

また、トンネル終端点間の遅延計測結果は各トンネル終端点が経路選択を行う際に必要となる。そのため、トンネル終端点は遅延計測結果を遅延データベースに登録後、遅延結果をトンネル終端点一覧に登録されている全トンネル終端点に通知する。通知には遅延データメッセージが用いられる。遅延データメッセージには遅延計測メッセージの送信元と送信先、及び、遅延の計測結果が含まれる。共通ヘッダー部分の Type フィールドには LATENCYDATA(60)、Source Node Address と Source Node Port フィールドには遅延データメッセージを送信するトンネル終端点の IP アドレスとポート番号を挿入する。遅延データ部分の Ping Source Node Address フィールドと Ping Source Node Port フィールドには遅延計測メッセージを送信したトンネル終端点の IP アドレスとポート番号、Ping Destination Node Port フィールドと Ping Destination Node Address フィールドには遅延計測メッセージを受信したトンネル終端点の IP アドレスとポート番号、Latency Data フィールドには遅延の計測結果を挿入する。この遅延データメッセージを受信したトンネル終端点は遅延データベースに指定されたトンネル終端点間の遅延データを登録する。そして、収集された遅延データを用いて遅延が最も小さい経路を選択する。

3.6 実装

本節では、LEON の実装について述べる。本節では、本実装のことを leond と呼ぶ。

3.6.1 実装環境

表 3.1 に本提案手法を実装するにあたり、利用した環境を示す。

表 3.1: 実装環境

CPU	Intel Xeon L5520 2.27Ghz * 2 (8 Cores)
RAM	24GB DDR3 RAM
NIC	Intel 82575EB Gigabit NIC
使用 OS	Debian 6.0.6 Squeeze
使用カーネル	Linux Kernel 3.7.2
使用言語	C 言語

3.6.2 実装概要

表 3.2 に leond の各機構を構成するモジュールを示す。また、各モジュール間の関係について図 3.15 に示す。

表 3.2: leond のモジュール

機構	モジュール
フレーム転送機構 終端点管理機構	main_poller
遅延計測機構	pinger
トポロジ構築機構	topology
経路表	fdb nodelist

leond の引数と起動例を下記に示す。

```
$ ./leond
--debug: debug mode
-d: run as daemon
-u: localaddr:port
-p: peeraddr:port
```

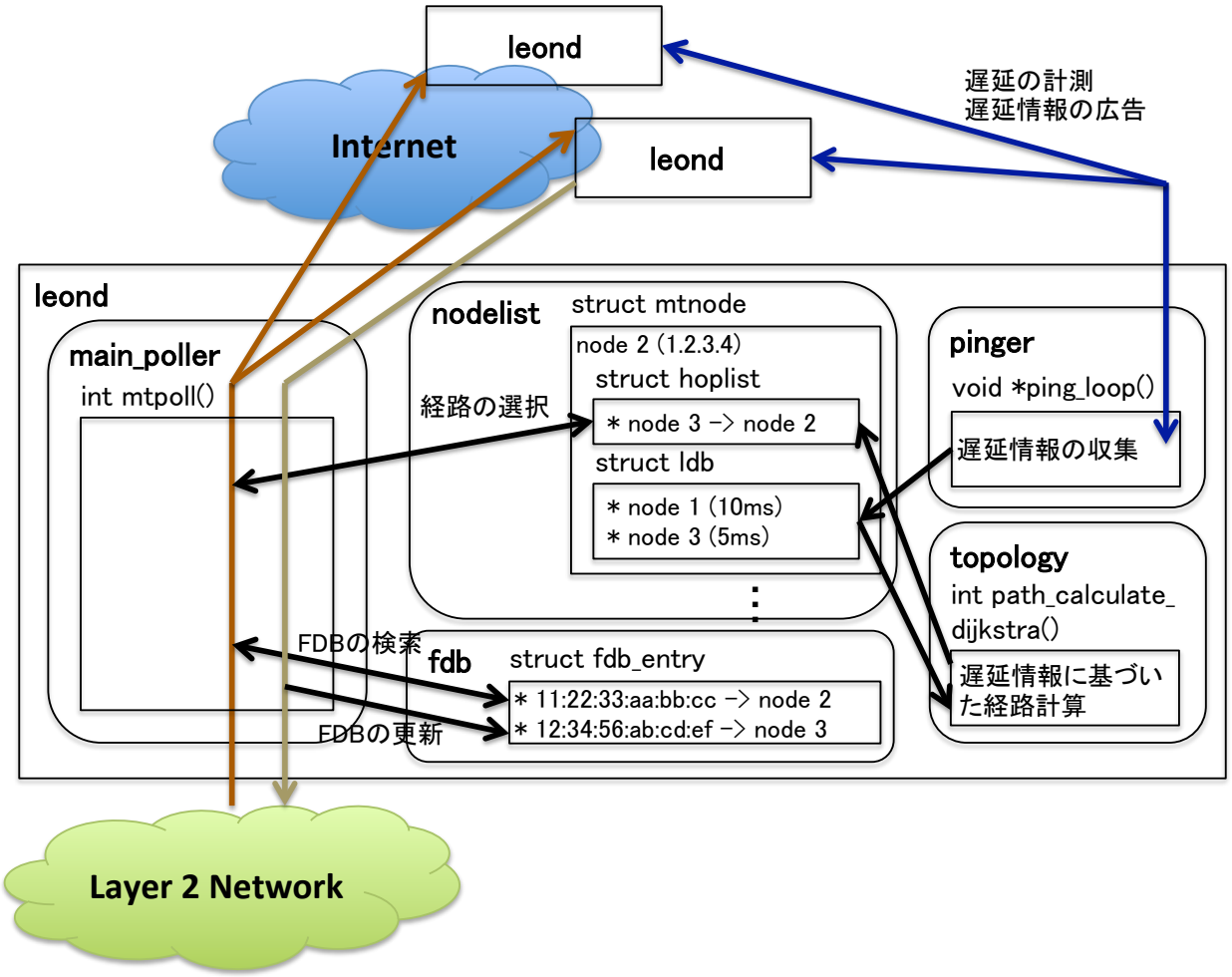

図 3.15: leond のモジュール間の関係

```
--ping-interval: ping interval seconds
```

```
$ ./leond -d -u 1.2.3.4:10000（既存の Layer 2 ネットワークに参加しない場合）
$ ./leond -d -u 5.6.7.8:10000 -p 1.2.3.4:10000（既存の Layer 2 ネットワークに
参加する場合）
```

leond を起動するにあたり、必須である引数は-u オプションのみである。-u オプションには他のトンネル終端点からの通信を待ち受けるために利用する IP アドレスとポート番号を指定する。また、既に他のトンネル終端点によって拡張されている Layer 2 ネットワークに参加する際には、既に参加しているトンネル終端点の IP アドレスとポート番号を-p オプションで指定する。

また、遅延の計測を行う間隔のデフォルト値は 60 秒にした。デフォルト値から変更したい場合は、--ping-interval オプションで遅延を計測する間隔を秒単位で指定することにより変更が可能である。デフォルト値を 60 秒に設定した理由は、遅延の計測を小さ

い間隔で行うと、遅延データメッセージと遅延計測メッセージが大量に発生してしまう可能性があったからである。Layer 2 ネットワークに参加しているトンネル終端点の台数を N、1 回の遅延計測で発生するメッセージ数を M とし、この 2 つの関係を数式で表すと次のようになる。

$$M = (2 + N) \times N$$

そのため、例えば 30 台のトンネル終端点で、遅延計測の間隔を 30 秒とすると、30 秒間で $M = (2 + 30) \times 30 = 960$ 個のメッセージが発生する。この内、1 台のトンネル終端点が処理するメッセージの量は $960 \div 30 = 32$ 個である。32 個のメッセージを 30 秒で処理するためには、トンネル終端点は 1 秒に 1 個以上処理する必要がある。これではトンネル終端点が高負荷状態となり、イーサネットフレームの転送に遅延が発生する可能性があったので、遅延計測の間隔を 60 秒とした。30 台のトンネル終端点で、遅延計測の間隔が 60 秒の場合、トンネル終端点はメッセージを 2 秒に 1 個以上処理すれば良い。

3.6.3 遅延計測機構

遅延計測機構は pinger モジュールによって構成されている。pinger モジュールは、トポロジー構築機構に他トンネル終端点にイーサネットフレームを転送するための経路を計算する際に必要となる情報を提供する。pinger モジュールの機能は 3 つある。以下にそれぞれについて述べる。

まず 1 つ目の pinger モジュールの機能として、遅延データベースの管理が挙げられる。leond を利用して拡張された Layer 2 ネットワークに参加しているトンネル終端点の情報は、後述する nodelist モジュールが提供する `mtnode` 構造体に記憶されている。`mtnode` 構造体は参加しているトンネル終端点に対して 1 つ存在する。pinger モジュールは `mtnode` 構造体の内部にある、`ldb` 構造体の管理を行う。`ldb` 構造体にはトンネル終端点間の遅延情報が記憶されている。pinger モジュールは遅延の計測結果、及び、他トンネル終端点から広告された遅延情報を利用し `ldb` 構造体の更新を行う。

2 つ目の pinger モジュールの機能として他トンネル終端点までの遅延計測が挙げられる。遅延計測機能は起動時に`--ping-interval` オプションによって指定された時間毎に呼び出される。起動時に指定されなかった場合は 60 秒毎に呼び出される。遅延計測機能が呼び出されると、 3.5.3 項で説明した、遅延計測メッセージを拡張された Layer 2 ネットワークに参加している全てのトンネル終端点に送信する。そして、他トンネル終端点から送信した遅延計測メッセージの返答を受け取ると、送信してから返信を受け取るまでに経過した時間を計算し、その時間を遅延として遅延データベースに登録する。また、遅延情報は他のトンネル終端点が経路計算を同様に行うために必要となるため、全てのトンネル終端点に遅延情報を 3.5.3 項で説明した、遅延データメッセージを用いて広告する。遅延データメッセージを受信したトンネル終端点は同様に、遅延情報を遅延データベースに登録する。

そして 3 つ目の pinger モジュールの機能として、トンネル終端点の死活監視が上げられる。あるトンネル終端点から遅延計測メッセージの返信がない場合、そのトンネル終端

点の遅延計測失敗回数を増加させる。失敗回数が 2 回になると、そのトンネル終端点で障害が発生したと判断し、そのトンネル終端点をトンネル終端点リストから消去する。そして、トポロジー構築機構を呼び出し、経路の再計算を行う。遅延計測失敗回数は、トンネル終端点から返信を受け取ると 0 に戻る。

3.6.4 トポロジー構築機構

トポロジー構築機構は topology モジュールによって構成される。topology モジュールは、leond を利用して拡張された Layer 2 ネットワークに参加している全てのトンネル終端点までの遅延が最も小さい経路を計算する。フレーム転送機構は topology モジュールによって計算された経路を基に、イーサネットフレームの転送を行う。

topology モジュールは pinger モジュールの遅延計測機能が呼び出された後に呼び出される。topology モジュールが呼び出されると、pinger モジュールによって収集された遅延データベースを基にダイクストラ法を用いて、トンネル終端点から拡張された Layer 2 ネットワークに参加している全てのトンネル終端点までの遅延が最も小さい経路の計算を行う。そして、経路はフレーム転送機構が利用できるように、`mtnode` 構造体の内部にある `hoplist` 構造体に記憶する。topology モジュールは計算した経路を、経由するトンネル終端点の `mtnode` 構造体へのポインターを、`hoplist` 構造体に経由する順番通りにリスト構造状で記憶する。

3.6.5 経路表

経路表は fdb モジュールと nodelist モジュールによって構成される。nodelist モジュールは同一の拡張された Layer 2 ネットワークに参加しているトンネル終端点の情報を管理する。トンネル終端点の情報は nodelist モジュールが管理する `mtnode` 構造体に格納される。`mtnode` 構造体にはトンネル終端点の IP アドレスとポート番号、遅延データベースである `ldb` 構造体へのポインターとトンネル終端点へイーサネットフレームへ転送するための経路情報が記憶されている `hoplist` 構造体へのポインターが格納されている。他のモジュールがトンネル終端点の情報の追加や削除、検索を行う際には、nodelist モジュールが提供する API を利用する。

また、fdb モジュールは拡張された Layer 2 ネットワーク内の各ホストがどのトンネル終端点によって収容されているかを学習する。fdb モジュールは他のトンネル終端点から転送されたイーサネットフレームを受信した際に呼び出される。fdb モジュールが呼び出されると受信したイーサネットフレームの送信元ホストの `fdb_entry` 構造体が存在するか検索する。`fdb_entry` 構造体が存在しない場合は、新しい `fdb_entry` 構造体を作成し、その送信元ホストと転送したトンネル終端点の `mtnode` 構造体へのポインターを記憶する。`fdb_entry` 構造体が存在する場合は、情報の更新が必要か確認し、必要の場合は更新を行う。また、fdb モジュールは他のモジュールから `fdb_entry` 構造体の検索、更新、作成を行うための API を提供する。フレーム転送機構がイーサネットフレームの転送を行う際には、fdb モジュールが提供する API を利用する。

3.6.6 フレーム転送機構と終端点管理機構

フレーム転送機構と終端点管理機構は main_poller モジュールによって構成されている。main_poller モジュールは他トンネル終端点から受信した通信の処理、及び、トンネル終端点が収容しているホストから受信したイーサネットフレームの転送処理といった外部からのイベント処理を行なっている。main_poller モジュールは Linux の I/O イベント通知機能である epoll を利用しイベントを待ち受ける。そして、イベントが発生すると mail_poller モジュールの処理が開始する。

発生したイベントがトンネル終端点が収容しているホストからのイーサネットフレーム受信である場合、イーサネットフレームの転送処理を行う。収容しているホストからイーサネットフレームを受信すると、イーサネットフレームの判別を行う。イーサネットフレームがブロードキャストフレームの場合は、全てのトンネル終端点に受信したイーサネットフレームを転送する。ユニキャストフレームの場合は、送信先ホストを fdb モジュールが提供している API を利用して検索し、転送先のトンネル終端点を決定する。そして、nodelist モジュールが管理している `mtnode` 構造体内の `hoplist` 構造体に従って、3.5.2 項で説明した転送メッセージのパケットを生成する。最後に、生成した転送メッセージを `hoplist` 構造体の一番最初に指定されているトンネル終端点へ転送する。

発生したイベントが他のトンネル終端点からの通信の場合、まず受信したメッセージの判別を行う。受信したメッセージが転送メッセージの場合、転送メッセージの転送経路情報を確認する。受信した転送メッセージが自身宛の転送メッセージの場合は、それを宛先ホストが収容されている Layer 2 ネットワークへ転送する。それ以外の場合は、転送経路情報で指定されたトンネル終端点へ転送メッセージを転送する。また、受信したメッセージがトンネル終端点の参加や離脱などを行うために利用されるコントロールメッセージの場合は、nodelist モジュールが提供する API を利用し、nodelist モジュールの `mtnode` 構造体を更新する。

第4章　評価

本章では、本研究の提案手法である LEON の評価を行う。

4.1　評価方針

3.3 節で示した通り、機能要件は、1. インターネット上に分散された複数の拠点への Layer 2 ネットワークの拡張、2. 宛先に応じた転送先の選択、3. 遅延の最も小さい経路でのイーサネットフレームの転送、4. 分散して動作すること、の 4 つである。機能要件について、LEON と 2.1 節と 2.2 節で挙げた既存研究が、どの程度満たしてるかを表 4.1 に示す。

表 4.1: LEON と既存研究の比較

機能要件	GRE/L2TP	VXLAN	N2N	LEON
1. インターネット上の複数環境への拡張	×	×	○	○
2. 宛先に応じた転送先の選択	×	○	○	○
3. 遅延の最も小さい経路での転送	×	×	×	○
4. 分散して動作する	×	○	×	○

GRE や L2TP などといった一対一型の Layer 2 ネットワーク拡張技術は、どの機能要件も満たしていない。一方、VXLAN と N2N は一対多型の Layer 2 ネットワーク拡張技術のため、両者共に機能要件の 2 は満たしている。しかし、両者共に遅延の最も小さい経路でイーサネットフレームを機能を転送する機能がないため、機能要件の 3 を満たしていない。インターネット上に分散した複数の拠点に Layer 2 ネットワークを拡張する LEON は、トンネル終端点間の遅延を計測し、計測結果から遅延の最も小さい経路を計算することができる。そして、イーサネットフレームを転送する際に、遅延の最も小さい経路で転送することができる。そのため、機能要件の 1、2、3 を満たす。また、N2N は Supernode を必要とするため、機能要件の 4 を満たさない。VXLAN と LEON は分散して動作するため、機能要件の 4 を満たす。以上より、LEON は機能要件を全て満たしている。

LEON はインターネット上に分散した複数の拠点を用いて展開されたサービスのパフォーマンスを向上させることを目標としている。これを実現するために、LEON は Layer 2 ネットワークのイーサネットフレームを遅延の最も小さい経路で転送をする。これにより、サービスを構成するコンポーネントの通信パフォーマンスが向上し、サービスのパフォーマン

スが向上すると予想される。そこで本研究では、LEON が与えるサービスのパフォーマンスへの影響を評価する。

しかし、一方で LEON は全てのトンネル終端点が、全てのトンネル終端点までの遅延計測や死活監視を行なっている。また、ブロードキャストフレームやコントロールメッセージを全てのトンネル終端点に転送する。そのため、トンネル終端点の台数が増加すると LEON に悪影響を与えると予想される。そこで本研究では、トンネル終端点の増加が LEON に与える影響の評価も行う。

4.2 サービスのパフォーマンス

本研究ではまず、LEON が遅延の最も小さい経路でイーサネットフレームを転送することによって、サービスのパフォーマンスに与える影響を評価する。本評価を行うにあたり、インターネット上に分散された複数の拠点を用いて構築されたサービスの例として WIDE Cloud を用いる。

WIDE Cloud には、 1.3.2 項で説明したように、複数の拠点でサービスを構築した場合、コンポーネントのパフォーマンスが低下するためサービスのパフォーマンスが低下してしまうという問題がある。低下するパフォーマンスの 1 つとして、仮想マシンのディスクパフォーマンスの低下が挙げられる。LEON を利用することにより、ストレージとの通信を行う際の遅延が小さくなるため、ストレージのパフォーマンスが改善されると予想される。これによって仮想マシンのディスクパフォーマンスが改善されると予想される。

本評価を行うにあたり、以下の 2 点の計測項目を評価に用いる。

- NFSv3 のファイルシステムパフォーマンス
- 仮想マシン内での Linux Kernel コンパイル所要時間

LEON は遅延の最も小さい経路でイーサネットフレームの転送を行う。そのため、直接通信を行った場合と比べ、LEON を利用した場合の方が小さい遅延で NFSv3 の通信を行うことができる。これにより、NFSv3 のパフォーマンスは直接通信した場合と LEON を利用した場合と比べると、LEON を利用した場合の方がファイルシステムのパフォーマンスが良いと予想される。本評価では、この予想が正しいかを知るため、NFSv3 のパフォーマンスを直接通信行った場合と LEON を利用した場合で計測し、これを評価をするための 1 つの指標とする。

また、NFSv3 のパフォーマンスが改善されることにより、仮想マシンのパフォーマンスも改善させると予想される。そこで、本評価では同様に、NFSv3 の通信を直接行った場合と LEON を利用した場合で、仮想マシン内での Linux Kernel [37] コンパイルにかかる時間を計測する。そして、これを評価をするための 1 つの指標とする。

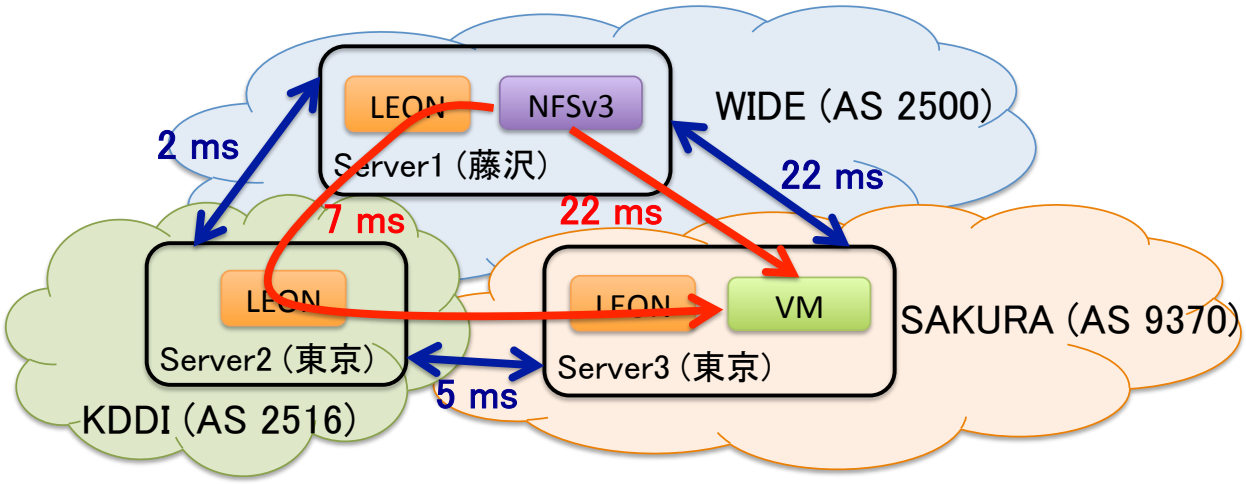

図 4.1: サービスパフォーマンス計測に用いた実験環境

4.2.1 実験環境

評価を行うにあたり、Layer 2 ネットワーク上で動作するサービスとそれを構成するコンポーネントのパフォーマンスを計測する。この計測を行うにあたって、実験環境に対して以下の要求が受けられる。

- 実インターネットでの計測環境
- 直接通信するよりも小さい遅延で通信することができる経路が存在すること

LEON はインターネット上に分散された複数の拠点に同一の Layer 2 ネットワークを拡張するために利用される。このような想定環境で構築されたサービスのパフォーマンスに与える影響を計測するため、本研究では実験環境を実インターネット上に構築し計測を行う。また、本研究では、最も小さい経路でイーサネットフレームを転送することによる影響を計測するため、実験環境には直接通信するより、他の拠点を経由することにより小さい遅延で通信することができる経路が存在する必要がある。

実験環境に対する要求から、本実験を行うにあたり、インターネット上に分散された 3 つの拠点に設置された 3 台のサーバーを利用して実験環境を構築した。この実験環境のトポロジー図を図 4.1 に示す。また、利用した実機サーバーの仕様を表 4.2 に示す。更に、それぞれのサーバーで利用したソフトウェアのバージョンを表 4.3 に示す。Server 1 は WIDE Project の藤沢 NOC に設置されたサーバーである。Server 2 は KDDI ウェブコミュニケーションズ社 [38] が提供している CloudCore VPS サービス [39] を利用した仮想サーバーである。そして、Server 3 は EditNet 社 [40] によるフレッツ光ネクスト接続サービス [41] [42] を用いて接続されたサーバーである。EditNet 社のサービスはさくらインターネット社 [43] のバックボーンを利用している [44]。

実験を行うにあたり、これら 3 台のサーバーで LEON の実装である leond を動作させた。WIDE Project とさくらインターネットは堂島でピアリングを行なっている。そのた

め、Server 1 と Server 3 が直接通信した場合、堂島を経由するために遅延が 22 ms 以上かかる。一方、WIDE Project と KDDI は東京でピアリングを行なっている。また、KDDI とさくらインターネットも同様に、東京でピアリングを行なっている。そのため、Server 1 と Server 3 が通信を行うには Server 2 を経由することにより、直接通信した場合よりも小さい遅延で通信をすることができる。leond はこの経路を自動的に発見し、Server 1 から Server 3 宛のイーサネットフレームを転送する際には Server 2 を経由させ転送をする。

表 4.2: サービスパフォーマンス計測に用いたサーバーの仕様

ホスト	CPU	メモリー	NIC
Server 1 (藤沢)	Intel Xeon L5520 2.27GHz	12GB	NetXtreme II BCM5709
Server 2 (東京)	AMD Phenom 9550 2.20GHz	2GB	virtio
Server 3 (東京)	Intel i7 870 2.93GHz	16GB	Intel 82574L

表 4.3: サービスパフォーマンス計測に用いた各サーバーのソフトウェアバージョン

サーバー	OS	Linux Kernel	gcc
Server 1 (藤沢)	Debian GNU/Linux 6.0.6 Squeeze	3.7.2	4.4.5
Server 2 (東京)	Fedora 17	3.7.4-204	4.7.2
Server 3 (東京)	Fedora 15	2.6.43.8-1	4.6.3

これにより、3 台のサーバーを用いて、本実験を行うための要求を満たした実験環境を構築した。

4.2.2 実験内容

本評価を行うにあたり、NFSv3 のファイルシステムパフォーマンスと仮想マシン内での Linux Kernel コンパイル所要時間を計測した。本実験は、NFSv3 の通信を直接インターネット上で行った場合と、LEON を利用して構築された Layer 2 ネットワーク上で行った場合の 2 通りで計測した。本実験を行うにあたり、 4.2.1 節で説明した実験環境において、Server 1 をストレージサーバー、Server 3 を仮想マシンを動かすサーバーとした。Server 1 のディスク領域を NFSv3 を利用して Server 3 から利用できるようにした。

NFSv3 のファイルシステムパフォーマンスの計測には Bonnie++ [45] を用いた。Bonnie++は Russel Coker 氏によって開発されたディスク・ファイルシステムパフォーマンス計測ツールである。本実験では、Server 3 上で Bonnie++を動作させ、NFSv3 のシーケンシャルアクセスパフォーマンスとランダムアクセスパフォーマンスを計測した。

また、仮想マシン内での Linux Kernel コンパイル所要時間を計測するために、Server 3 上で仮想マシンを動作させた。仮想マシンを動作させるための仮想化技術には Kernel-based Virtual Machine(=KVM) [46] を利用した。また、仮想マシンのディスクイメージ

は Server 1 に記憶されており、NFSv3 を経由して仮想マシンに提供される。仮想マシンの内部で Linux Kernel のコンパイルを行い、Linux Kernel のコンパイルにかかる時間を time コマンドを用いて計測した。

4.2.3 実験結果

サービスのパフォーマンス計測としてまず、NFSv3 のファイルシステムパフォーマンスを計測した。計測はインターネット上の経路で直接 NFSv3 の通信を行った場合と LEON を利用して拡張された Layer 2 ネットワーク上で NFSv3 の通信を行った場合の 2 通りで計測を行った。シーケンシャルアクセスパフォーマンスの計測結果を表 4.4、ランダムアクセスパフォーマンスの計測結果を表 4.5 に示す。計測はそれぞれ 3 回行い、計測結果は 3 回の平均値である。

表 4.4: NFSv3 のシーケンシャルアクセスパフォーマンス

通信手法	read (K/sec)	write (K/sec)
Direct	4537	5372
LEON	4726	5198

表 4.5: NFSv3 のランダムアクセスパフォーマンス

通信手法	seek(/sec)	create(/sec)	info(/sec)	delete(/sec)
Direct	2985	23	46	46
LEON	8417	65	132	132

インターネット上の経路で直接通信を行った場合のシーケンシャルアクセスパフォーマンスは、読み込み速度が 4537 K/sec、書き込み速度が 5372 K/sec であった。一方、LEON を利用して拡張された Layer 2 ネットワーク上で通信を行った場合のシーケンシャルアクセスパフォーマンスは、読み込み速度が 4726 K/sec、書き込み速度が 5198 K/sec であった。この計測結果から、シーケンシャルアクセスのパフォーマンスは、インターネット上の経路で直接通信を行った場合と、LEON を利用して拡張された Layer 2 ネットワーク上で通信を行った場合では、大きは変化はないということがわかった。

一方、インターネット上の経路で直接通信を行った場合のランダムアクセスパフォーマンスは、seek 操作が毎秒 2985 回、create 操作が毎秒 23 回、info 操作が毎秒 46 回、delete 操作が毎秒 46 回という結果となった。LEON を利用して拡張された Layer 2 ネットワーク上で通信を行った場合では、seek 操作が毎秒 8417 回、create 操作が毎秒 65 回、info 操作が毎秒 132 回、delete 操作が毎秒 132 回であった。この計測結果から、ランダムアクセスのパフォーマンスは、インターネット上の経路で直接通信を行った場合に比べ、LEON

表 4.6: Kernel コンパイルの所要時間

通信手法	コンパイル所要時間 (分)
Direct	78.9
LEON	58.8

を利用して拡張された Layer 2 ネットワーク上で通信を行った場合では約 3 倍のパフォーマンスとなることがわかった。

次に、サービスのパフォーマンス計測として、仮想マシン内での Linux Kernel のコンパイル所要時間を計測した。その計測結果を表 4.6 に示す。インターネット上の経路で直接通信を行った場合のコンパイル所要時間は 78.9 分であった。一方、LEON を利用して拡張された Layer 2 ネットワーク上で通信を行った場合のコンパイル所要時間は 58.8 分であった。LEON を利用して拡張された Layer 2 ネットワーク上で通信を行うことにより、直接通信を行った場合と比べ、コンパイル所要時間を約 20 分削減できることがわかった。

4.2.4 考察

本研究では、LEON がサービスのパフォーマンスに与える影響を評価した。本評価を行うために、インターネット上の経路で直接通信した場合と、LEON を利用して拡張された Layer 2 ネットワーク上で通信を行った場合の NFSv3 のファイルシステムパフォーマンスと仮想マシン内での Linux Kernel コンパイル所要時間を計測した。

LEON を利用することによりランダムアクセスパフォーマンスは改善されたが、シーケンシャルアクセスパフォーマンスは改善されなかった。NFSv3 のシーケンシャルアクセスのパフォーマンスはサーバー間の帯域に依存すると考えられる。今回構築した実験環境では、フレッツ光ネクスト接続サービスを用いてインターネットに接続されたサーバーの帯域が細いため、このサーバーの回線が帯域のボトルネックとなっている。そのため、シーケンシャルアクセスのパフォーマンスは改善されなかったと考えられる。一方、NFSv3 のランダムアクセスのパフォーマンスはサーバー間の遅延に依存すると考えられる。NFSv3 において 1 秒間に行えるランダムアクセスの回数は、1 秒間にサーバー間で何回 NFSv3 のパケットをやり取りできるかに比例する。LEON を利用することにより、サーバー間の遅延は小さくなる。そのため、1 秒間にサーバー間でやり取りできるパケット数が大きくなったため、ランダムアクセスのパフォーマンスが改善されたと考えられる。

NFSv3 のランダムアクセスのパフォーマンスが改善されることにより、仮想マシン内での Linux Kernel コンパイル所要時間は約 20 分短縮された。仮想マシン内での Linux Kernel コンパイル作業では多くのランダムアクセスが生じると予想される。そのため、NFSv3 のランダムアクセスのパフォーマンスが改善されたため、仮想マシン内でより高速なランダムアクセスが可能になったため、Linux Kernel コンパイル所要時間が短縮されたと考えられる。

本実験の実験結果から、LEON を利用することにより、サービスのパフォーマンスは改

善されるということがわかった。LEON は遅延の最も小さい経路でイーサネットフレームの転送を行う。これにより、直接通信した場合と比べ、小さい遅延での Layer 2 ネットワーク上のコンポーネント間の通信を可能とする。そのため、コンポーネントのパフォーマンスが改善され、サービスのパフォーマンスが改善される。

4.3 トンネル終端点の増加による影響

本研究では次に、トンネル終端点の増加が LEON に与える影響の評価を行う。LEON を利用して Layer 2 ネットワークを拡張しているトンネル終端点の台数が増加すると様々な影響が生じると考えられる。生じる影響の 1 つとして中継を行なっているトンネル終端点において障害が発生してから、通信が復旧するまでの時間の増加が考えられる。本研究では、このような障害が発生してから通信復旧までにかかる時間に着目した。

LEON は宛先のトンネル終端点に、遅延の最も小さい経路で、イーサネットフレームを転送する。これを行うため、LEON では、 3.5.3 項で説明したような遅延データベースを構築する。そして、遅延データベースを用いて、Layer 2 ネットワークに参加している 1 つ 1 つのトンネル終端点までの遅延が最も小さくなる経路を事前に計算し、拡張された Layer 2 ネットワークのトポロジーを作成している。イーサネットフレームはこのトポロジーに基いて転送される。

VXLAN と N2N は遅延の最も小さい経路でイーサネットフレームの転送を行わないため、拡張された Layer 2 ネットワークのトポロジーを作成しない。トポロジーを作成しない利点として、Layer 2 ネットワークに参加している何れかのトンネル終端点で障害が発生しても、拡張された Layer 2 ネットワークにおいて障害は発生しないという点が挙げられる。VXLAN の場合、何れかのトンネル終端点で障害が発生しても、IP マルチキャストが正常に動作していれば Layer 2 ネットワーク上での通信は正常に行える。同様に、N2N の場合でも、Supernode で障害が発生しなければ Layer 2 ネットワーク上での通信は正常に行える。そのため、トポロジーを作成しない VXLAN と N2N では、あるトンネル終端点で発生した障害によって拡張された Layer 2 ネットワークが影響を受けることはない。

一方で、遅延の最も小さい経路でイーサネットフレームの転送を行うためにトポロジーを作成する LEON は、ある終端点で発生した障害によって拡張された Layer 2 ネットワークが一時的に影響を受ける場合がある。これは、障害が発生したトンネル終端点が、イーサネットフレームの中継を行なっていた場合ある。中継を行なっているトンネル終端点で障害が発生すると、そのトンネル終端点を経由する経路が全て利用できなくなる。LEON は遅延計測を行うと同時に、トンネル終端点の死活監視も行なっている。障害を検知すると、障害が発生しているトンネル終端点をトンネル終端点リストから削除し、経路の再計算を行う。しかし、LEON ではデフォルトで遅延計測を 60 秒に 1 回で行なっていて、遅延計測メッセージの応答が 2 回ない場合に障害発生と判断している。そのため、障害が発生から検知までに最大 120 秒を必要とする。検知してトポロジーが収束するまでは、障害が発生したトンネル終端点にイーサネットフレームを転送し続けるため、障害が発生したトンネル終端点を経由する通信は宛先に到達できなくなる。

また、LEON では拡張された Layer 2 ネットワークに参加している全てのトンネル終端点が、それぞれ異なるトポロジーを作成している。そのため、あるトンネル終端点で障害が発生してから Layer 2 ネットワーク上の通信が完全に正常に戻るには、全てのトンネル終端点が障害を検知し、トポロジーを作成し直す必要がある。つまり、障害が発生してから復旧までかかる時間は、Layer 2 ネットワークに参加しているトンネル終端点の台数の影響を受ける可能性があると考えられる。

そこで本評価では、Layer 2 ネットワークに参加しているトンネル終端点の台数が、あるトンネル終端点の障害が発生してから Layer 2 ネットワークの通信が正常化するまでかかる時間に与える影響を調査する。調査を行うために、トンネル終端点を徐々に増加させ、障害発生から全てのトンネル終端点でのトポロジー収束までかかる時間を計測する。具体的には、全てのトンネル終端点の通信を中継しているトンネル終端点で障害を発生させ、そのトンネル終端点を経由していた通信が復旧するまでにかかる時間を計測する。この計測をトンネル終端点の台数を変化させながら行う。

4.3.1 実験環境

本研究では、イーサネットフレームの転送を中継しているトンネル終端点で障害が発生した際に、Layer 2 ネットワークに参加しているトンネル終端点の台数が、トポロジーの収束時間に与える影響を評価する。評価を行うにあたり、Layer 2 ネットワークに参加しているトンネル終端点の台数を変化させ、それぞれの場合での障害発生から全てのトンネル終端点でトポロジーが収束するまでにかかる時間を計測する。この計測を行うにあたって、実験環境に対して以下の要求が受けられる。

- 全てのトンネル終端点間に遅延が存在すること
- 直接通信するよりも小さい遅延で通信することができる経路が存在すること
- 数十台のトンネル終端点

LEON はインターネット上に分散された複数の拠点に同一の Layer 2 ネットワークを拡張するために利用される。インターネットにおいて、分散された複数の拠点同士が通信するにあたり、遅延が生じる。そのため、実験環境を実インターネット環境と類似した環境にするためには、実験環境のトンネル終端点間に遅延が存在する必要がある。

また、本実験では、中継をするトンネル終端点で障害が発生してから、全てのトンネル終端点でトポロジーが収束するまでの時間を計測する。この計測を行うには、あるトンネル終端点がイーサネットフレームを転送する際に、中継するトンネル終端点が必要となる。LEON は直接宛先のトンネル終端点へ転送するよりも、他のトンネル終端点を経由して宛先のトンネル終端点へ転送することにより、小さい遅延で転送できるような経路が存在した場合に中継するトンネル終端点を設定する。そのため、実験環境では、直接通信するよりも小さい遅延で通信することができる経路が必要となる。

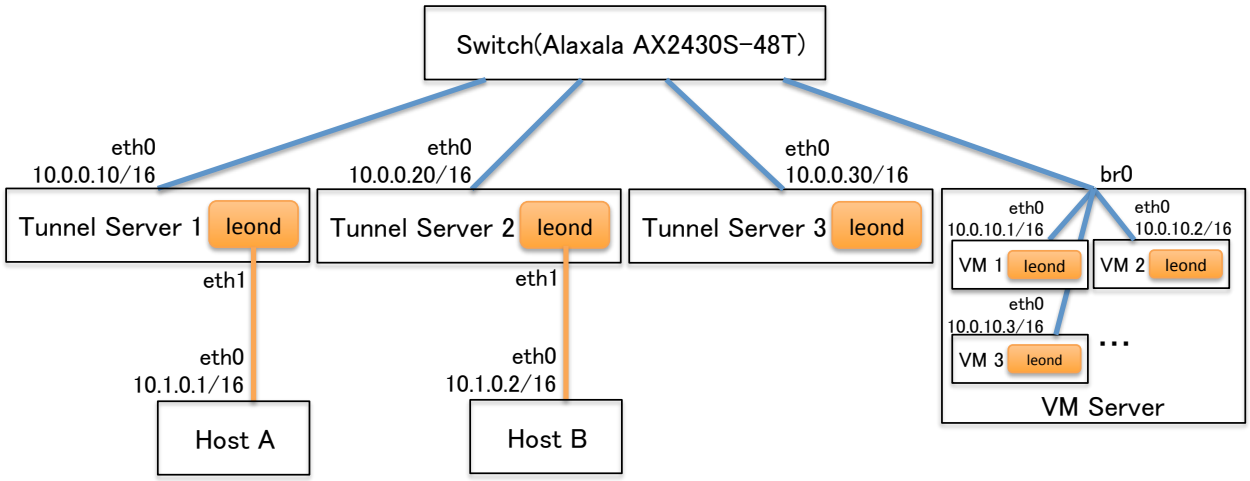

図 4.2: 実験環境のトポロジー図

さらに、 3.2 節で説明したように、LEON は数十の拠点に Layer 2 ネットワークを拡張することを想定している。想定環境と類似した環境で実験を行うため、実験は最大で数十台のトンネル終端点という規模で行う。

実験環境に対する要求から、本実験を行うにあたり、6 台の実機サーバーを利用して実験環境を構築した。この実験環境のトポロジー図を図 4.2 に示す。また、利用した実機サーバーの仕様を表 4.7 に示す。更に、それぞれのサーバーで利用したソフトウェアのバージョンを表 4.8 に示す。6 台の実機サーバーの内、3 台で LEON の実装である leond を動作させた。このうち 2 台に、実際に拡張された Layer 2 ネットワークに参加するホストサーバーとして実機サーバーを直接接続した。さらに、残りの実機サーバー 1 台で最大 27 台の仮想マシンを動作させ、仮想マシン内部で leond を動作させた。leond が動作するサーバーは全て同じ Layer 2 ネットワークに所属している。

表 4.7: 実験で利用した実機サーバーの仕様

ホスト	CPU	メモリー	NIC
Tunnel Server 1	Intel Xeon L5520 2.27Ghz	12GB	NetXtreme II BCM5709
Tunnel Server 2	Intel Xeon L5520 2.27Ghz	24GB	Intel 82575EB
Tunnel Server 3	Intel Xeon E5430 2.66Ghz	4GB	NetXtreme II BCM5708
Host A	Intel Xeon 5160 3.00Ghz	4GB	Intel 80003ES2LAN
Host B	Intel Xeon 5160 3.00Ghz	4GB	Intel 80003ES2LAN
VM Server	AMD Opteron 6128	20GB	Intel 82576

また、leond が動作するサーバー間で、tc(Linux Traffic Control) [47] を利用して擬似的に遅延を発生させた。サーバー間に設定した遅延を図 4.3 に示す。全てのトンネル終端点において、Tunnel Server 3 への遅延は他のサーバーへの遅延よりも小さく設定した。こ

表 4.8: 実験で利用した各サーバーのバージョン

サーバー	OS	Linux Kernel	gcc
Tunnel Server 1 Tunnel Server 2 Tunnel Server 3 VM #	Debian GNU/Linux 6.0.6 Squeeze	3.7.2	4.4.5
Host A Host B	Debian GNU/Linux 6.0.6 Squeeze	2.6.32-5	4.4.5
VM Server	Ubuntu 10.04.4 LTS	3.7.2	4.4.3

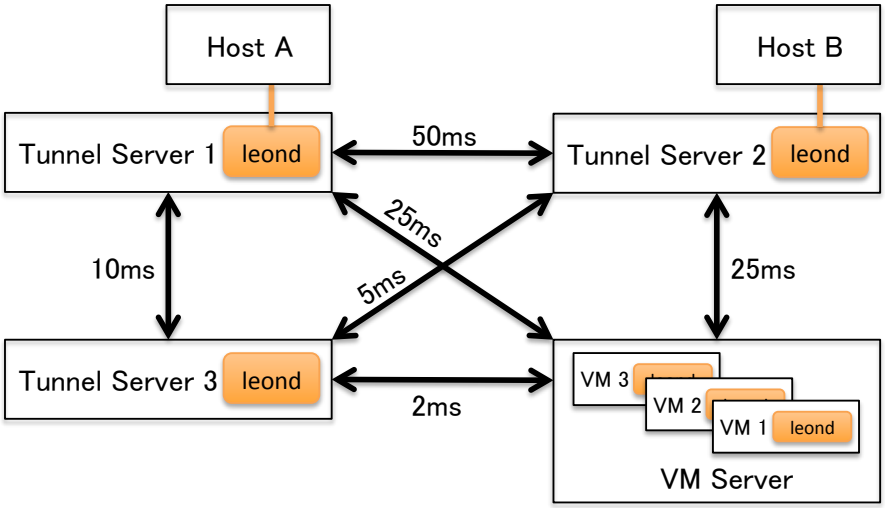

図 4.3: 実験環境に設定された擬似的な遅延

れにより、直接通信するよりも小さい遅延で通信することができる経路を作成した。全てのサーバー間の通信は Tunnel Server 3 を経由することで、直接通信するよりも小さい遅延で通信をすることができる。仮想マシン間には遅延を設定していない。

これによって、実機サーバー 6 台を用いて、本実験を行うための要求を満たした実験環境を構築した。

4.3.2 実験内容

本実験では、中継するトンネル終端点で障害が発生してから、全てのトンネル終端点でトポロジーが収束するまでの時間を計測する。中継をするトンネル終端点で障害が発生すると、そのトンネル終端点を経由して行われていた通信は、トポロジーが収束するまで行えなくなる。本実験では、この性質を利用してトポロジーの収束時間を計測するプログラムを作成した。本プログラムは、あるトンネル終端点から全てのトンネル終端点に対して

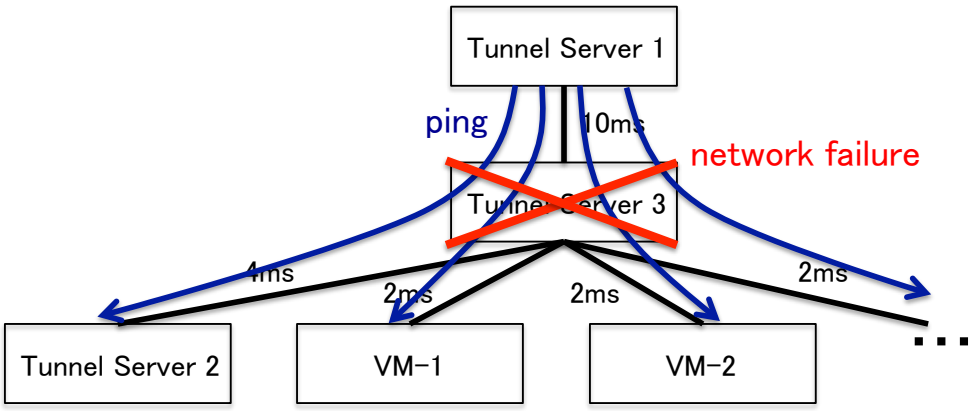

図 4.4: Tunnel Server 1 で構築されるトポロジーと実験の様子

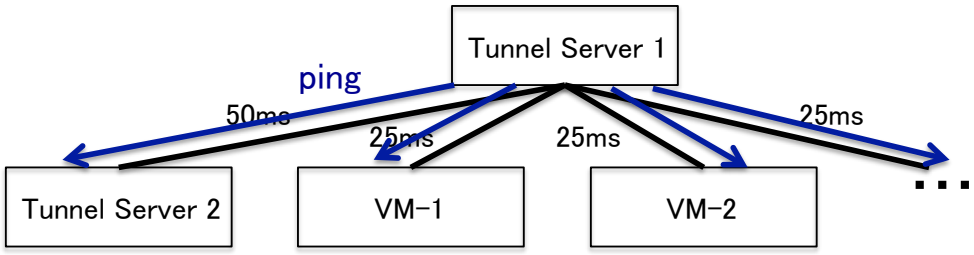

図 4.5: トポロジー収束後のトポロジー

定期的に ICMP Echo Request を送信し、その返信を監視する。本プログラム開始後、中継を行なっているトンネル終端点でネットワーク障害を発生させると、全てのトンネル終端点から返信がなくなる。返信がなくなると、中継するトンネル終端点で障害が発生したと判断し計測を開始する。そして、全てのトンネル終端点でトポロジーが収束すると、全てのトンネル終端点から返信を受信できるようになる。本プログラムが全てのトンネル終端点から返信を受け取ると、全てのトンネル終端点でトポロジーが収束したと判断し計測を終了する。

作成したプログラムは Tunnel Server 1 で動作させた。実験環境下で Tunnel Server 1 で動作する leond が構築する Layer 2 ネットワークのトポロジーと実験の様子を図 4.4 に示す。Tunnnel Server 1 が、他のトンネル終端点へイーサネットフレームを転送する際には、必ず Tunnel Server 3 を経由する。作成したプログラムを Tunnel Server 1 で開始後、Tunnel Server 3 のネットワーク接続を切断した。Tunnel Server 3 が切断されると、全てのトンネル終端点は Tunnel Server 1 と通信ができなくなる。全てのトンネル終端点で動作する leond が、Tunnel Server 3 での障害を検知すると、トポロジーが再構築され、再び Tunnel Server 1 と通信ができるようになる。障害検知後の Tunnel Server 1 で動作する leond が構築するトポロジーは図 4.5 で示すようなトポロジーとなる。Tunnel Server 1 と全てのトンネル終端点が通信可能になった時点で計測終了である。

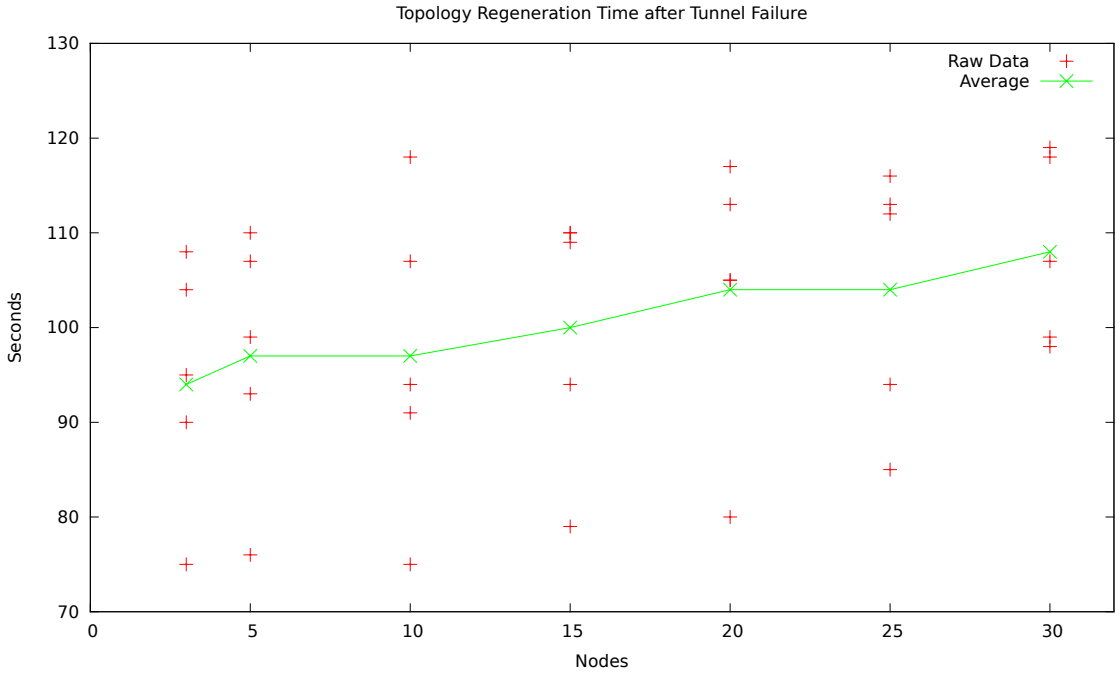

図 4.6: ノード数に応じたトポロジー収束時間

4.3.3 実験結果

本実験では、3 台から 30 台までのトンネル終端点を利用して拡張した Layer 2 ネットワークにおいて、中継を行なっているトンネル終端点での障害発生後、全てのトンネル終端点でトポロジーが収束するまでにかかった時間を計測した。その実験結果を図 4.6 に示す。本実験では、3 台から 30 台までの各トンネル終端点数について、 4.3.2 節で説明した計測を 5 回行った。計測を行う度に全てのトンネル終端点で leond を再起動し、Layer 2 ネットワークの再構築を行った。

図 4.6 から、中継を行なっているトンネル終端点で障害が発生してから全てのトンネル終端点でトポロジーが収束するまでにかかる時間は、同じトンネル終端点の台数でもばらつきが大きいことがわかる。25 台のトンネル終端点では、トポロジーの収束にかかった時間は最小で 81 秒、最大で 123 秒と 42 秒の差がある。一方で、同じ台数でもばらつきはあるが、各台数の平均値から、台数が増加するに連れトポロジーが収束するまでにかかる時間の平均値は緩やかに増加していることがわかる。そのため、Layer 2 ネットワークに参加しているトンネル終端点の台数が増加するに連れ、中継を行なっているトンネル終端点で障害が発生してから全てのトンネル終端点でトポロジーが収束するまでにかかる時間は少しずつ増加する、と考えられる。

4.3.4 考察

本研究では、トンネル終端点の増加による影響を評価した。そして本評価では、Layer 2 ネットワークに参加しているトンネル終端点の台数が、中継を行なっているトンネル終端点において障害が発生してから全てのトンネル終端点でトポロジーが収束するまでかかる時間に与える影響を調査した。調査をするにあたり、擬似的に遅延を発生させ、直接通信するよりも小さい遅延で通信できる経路が存在する実験環境を構築し、評価実験を行った。その結果、同じトンネル終端点の台数でも、トポロジーが収束する時間のばらつきは大きいが、各台数の平均値からトンネル終端点の台数が増加するに連れ、トポロジーの収束にかかる時間は緩やかに増加する、ということがわかった。よって、トンネル終端点の増加は LEON に悪影響を及ぼすということがわかった。

この原因は、トンネル終端点を起動した時間の差であると考えられる。あるトンネル終端点で、トンネル終端点の障害発生からトポロジーの再構築までかかる時間を T 秒、遅延計測を行う間隔時間を T_p 秒、最後に遅延計測を行なってから障害発生までに経過した時間を T_e 秒とすると、これらの関係を次の数式で表すことができる。

$$T = (T_p - T_e) + T_p$$

遅延計測を行う間隔時間 $(=T_p)$ はデフォルト状態で 60 秒である。LEON は遅延計測が 2 回失敗すると障害発生と判断するため、障害発生からトポロジーの再構築までかかる時間 $(=T)$ は、T_p 秒以上となる。例えば、最後に遅延計測を行なってから 20 秒経過した時点で障害が発生すると、それを検知しトポロジーの再構築までかかる時間は $T = (60-20)+60 = 100$ より 100 秒である。障害発生からトポロジーの再構築までかかる時間 $(=T)$ は、最後に遅延計測を行なってから障害発生までに経過した時間 $(=T_e)$ が小さいほど大きくなる。トンネル終端点が複数の場合、全てのトンネル終端点を同時に起動をしていないので、T_e の値にはトンネル終端点ごとにばらつきがある。トンネル終端点の台数が増えると T_e の値のばらつきが大きくなり、最後に遅延計測を行なってから障害発生までに経過した時間が短いトンネル終端点がいる確率が高まる。そのため、トンネル終端点の台数が増加するに連れ、中継を行なっているトンネル終端点において障害が発生してから全てのトンネル終端点でトポロジーが収束するまでかかる時間が増加すると考えられる。

RFC 793 では TCP のタイムアウト時間は 300 秒と定義されている [48]。第 2 章で挙げた Layer 2 ネットワーク拡張技術を利用して構築された Layer 2 ネットワークで障害が発生すると、管理者が障害の原因を特定し、修正するまでに時間がかかる。この手法では、障害が発生してから修正まで 300 秒以上かかり、TCP のセッションが切断されてしまう場合が多い。LEON では、中継を行なっているトンネル終端点で障害が発生してから、最大 120 秒で通信が再び可能となる。そのため、LEON を用いることで、障害発生時は TCP のセッションが切断される前に復旧することができる。

4.4 実験のまとめ

本研究では、インターネット上に分散した複数の拠点を用いて構築されたサービスのパフォーマンス改善を目標としている。従来手法では、複数拠点を用いてサービスを構築した場合、サービスを構成するコンポーネントのパフォーマンスが遅延により低下してしまうため、サービスのパフォーマンスが低下してしまうという問題があった。この問題を解決するため、LEON は遅延の最も小さい経路でイーサネットフレームの転送を行う。これにより、コンポーネントのパフォーマンスが改善され、サービスのパフォーマンスが改善されるということがわかった。

しかし、一方でトンネル終端点の台数が増加すると LEON に悪影響を及ぼすということがわかった。トンネル終端点の増加が与える影響として、中継を行うトンネル終端点で障害が発生してから、そのトンネル終端点を経由していた通信の復旧までにかかる時間の増加があるということもわかった。これは LEON が分散して動作するために、Layer 2 ネットワークに参加している全てのトンネル終端点が全てのトンネル終端点の管理、死活監視や遅延計測などを行う必要があるためであると考える。そのため、LEON は多くの拠点に同一の Layer 2 ネットワークを拡張するためには適していないということがわかった。

第5章　結論

本章では、本論文のまとめと今後の展望を示す。

5.1　本研究のまとめ

本研究では、インターネット上の複数の拠点に同一の Layer 2 ネットワークを拡張することができる一対多型の Layer 2 ネットワーク拡張技術である LEON の設計と実装をした。現在のインターネットには、宛先と直接通信した場合より、他の拠点を経由して宛先と通信した場合の方が、通信をする際の遅延が小さくなる場合がある。LEON はトンネル終端点間の遅延を計測し、その計測結果をもとに遅延が最も遅延が小さくなる経路を計算する。これにより、LEON を利用することにより、イーサネットフレームを遅延の最も小さい経路で転送できるようになった。その結果、本研究で行った実験から、LEON を利用して拡張された Layer 2 ネットワーク上で通信を行った場合、従来手法と比べ、サービスを構成するコンポーネントのパフォーマンスが改善され、サービスのパフォーマンスも改善されるということがわかった。しかし、LEON では全てのトンネル終端点が他のトンネル終端点の状態管理や遅延計測などを行う必要があるため、トンネル終端点が増加することによりトンネル終端点の負荷が高くなるため様々な悪影響が生じる。本研究で行った実験から、トンネル終端点の増加が与える影響の 1 つとして、中継を行なっているトンネル終端点での障害発生から、そのトンネル終端点を経由していた通信が再び可能になるまでかかる時間の増加があるということがわかった。そのため、LEON は多くの拠点に Layer 2 ネットワークを拡張するには適していないということもわかった。

5.2　今後の展望

本研究で提案した Layer 2 ネットワーク拡張技術である LEON は、中継を行なっているトンネル終端点で障害が発生してから、再びそのトンネル終端点を中継して行われていた通信が行えるようになるまで、最大で 120 秒かかる。そのため、最大 120 秒間、通信が行えない間に、送信されたイーサネットフレームがパケットロスされてしまう可能性やサーバーから切断されてしまうなどといった問題の原因となることが想定される。また、図 5.1 で示すように、ファイアーウォールの設定や拠点間のネットワーク障害などにより、直接通信することはできないが、他のトンネル終端点からは到達することができる場合が考えられる。LEON はこのよう場合、直接通信することができないトンネル終端点は障害発生と判断し、トンネル終端点リストから消去してしまう。そのため、Layer 2 ネット

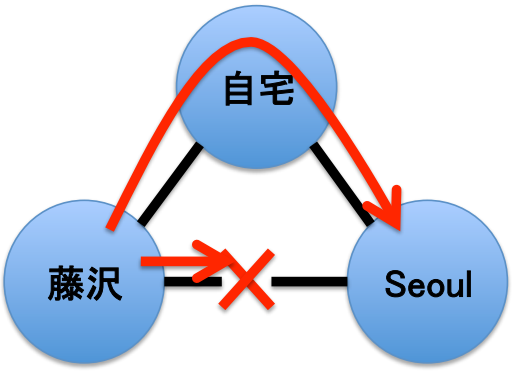

図 5.1: 直接通信をすることができない状況

ワークが分断してしまうという問題がある。これらの問題を解決するためには、より優れた障害検知の手法が必要である。

また、本研究で提案した手法は、トンネル終端点間の遅延を計測し、その計測結果をもとに経路の計算を行った。しかし、インターネットには遅延は小さいが帯域幅は細いトンネル終端点や、遅延は大きいが帯域幅は太いトンネル終端点が存在する可能性がある。WIDE Cloud のように拡張された Layer 2 ネットワーク上で様々なアプリケーションが動作するような環境では、遅延よりも帯域幅を優先したほうがパフォーマンスが高くなるアプリケーションも動作している可能性がある。そのため、Layer 2 ネットワーク拡張技術はトンネル終端点間の遅延だけでなく、Layer 2 ネットワーク上で動作するアプリケーションやトンネル終端点間の帯域幅も考慮する必要がある。

謝辞

本論文の作成にあたり、ご指導頂きました慶應義塾大学環境情報学部教授 村井純博士、同学部 教授中村修博士、同学部准教授 楠本博之博士、同学部准教授 Rodney D. Van Meter III 博士、政策・メディア研究科特任講師 吉藤英明博士、同研究科特任講師斉藤賢爾博士、同准教授 植原啓介博士に感謝致します。

研究について日頃からご指導頂きました政策・メディア研究科博士課程 堀場勝広氏に感謝致します。研究室に所属して以来、WIDE Cloud を始めとする様々なインターネットの研究や技術について教えて頂きました。また、研究に行き詰まった際に的確なアドバイス等頂きました。本研究を卒業論文としてまとめることが出来たのも堀場勝広氏のおかげです。重ねて感謝申し上げます。

また、研究について同じくアドバイスを頂きました政策・メディア研究科博士課程 岡田耕司氏、空閑洋平氏に感謝致します。特に岡田耕司氏には、研究だけでなく、文章の書き方やネットワーク運用に関するアドバイスを多く頂きました。重ねて感謝申し上げます。

さらに、同じくご指導頂きました独立行政法人情報通信研究機構 (NICT) 田崎創博士に感謝致します。研究室に所属して間もない頃、インターネットに関して初歩的なところから手取り足取り教えて頂きました。また、様々な研究活動に参加する機会を作って頂きました。WIDE Project に参加するきっかけを作って頂いたのも田崎創博士です。田崎創博士からは研究を始め、多くのことを学ばせて頂きました。重ねて感謝申し上げます。

研究室を通じた生活の中で多く示唆を与えてくださった政策・メディア研究科博士課程 松谷健史氏、修士課程 佐藤弘崇氏、三部剛義氏、横石雄大氏、環境情報学部上野幸杜氏、木本瑞希氏、三條場直希氏、鴻野弘明氏、中島明日香氏、倉田彩子氏、水谷伊織氏に感謝します。また、徳田・村井・楠本・中村・バンミーター・植原・三次・中澤・武田合同研究プロジェクトの皆様に感謝致します。

また、本論文を作成中、公私共にお世話になった青山学院大学理工学研究科修士課程 杉浦拓夢氏、東京農業大学農学部 宗像祥久氏、慶應義塾大学文学部 坪田未歩氏、同大学環境情報学部 永井俊行氏、村上孝太氏に感謝致します。諸氏には、本論文を作成するにあたり、多くの励ましやアドバイス等頂きました。諸氏なしでは本論文を完成させることができませんでした。重ねて感謝申し上げます。

以上をもって本論文の謝辞とさせていただきます。

参考文献

[1] Zone Research. The Need For Speed II. *Zone Market Bulletin, Issue 05*, 2001.

[2] Akamai Press Release. Akamai and Jupiter Research Identify '4 Seconds' as the New Threshold of Acceptability for Retail Web Page Response Times. Press Release, November 2006. http://www.akamai.com/html/about/press/releases/2006/press_110606.html.

[3] Erik Nygren and Ramesh K. Sitaraman. The Akamai Network: A Platform for High-Performance Internet Applications. *ACM SIGOPS*, pages 2–19, July 2010.

[4] Akamai Technologies. http://www.akamai.com/, December 2012.

[5] NTT Communications. http://www.ntt.com/, December 2012.

[6] NTT Communications Press Release. 東北電力の「計画停電」による影響について. Press Release, March 2011. http://www.ntt.com/release/monthNEWS/detail/20110315_2.html.

[7] 慶應義塾大学 湘南藤沢キャンパス（SFC）. http://www.sfc.keio.ac.jp/, December 2012.

[8] Datagram. http://www.datagram.com/, December 2012.

[9] Huffington Post. Websites Scramble As Hurricane Sandy Floods Data Centers. News Paper, October 10, 2012. http://www.huffingtonpost.com/2012/10/30/hurricane-sandy-websites-floods-data-centers_n_2046034.html.

[10] Internap. http://www.internap.com/, December 2012.

[11] PEER 1 Hosting. http://www.peer1.com/, December 2012.

[12] Data Center Knowledge. Massive Flooding Damages Several NYC Data Centers, October 30, 2012. http://www.datacenterknowledge.com/archives/2012/10/30/major-flooding-nyc-data-centers/.

[13] Amazon Web Services. http://aws.amazon.com/jp/, December 2012.

[14] Amazon Web Services. Overview of Security Processes. White Paper, May 2011. http://awsmedia.s3.amazonaws.com/pdf/AWS_Security_Whitepaper.pdf.

[15] Cogent Communications. http://www.cogentco.com/, December 2012.

[16] Sprint. http://www.sprint.com/, December 2012.

[17] PC World. Sprint-Cogent Dispute Puts Small Rip in Fabric of Internet, October 31, 2008. http://www.pcworld.com/article/153123/sprint_cogent_dispute.html.

[18] N.M. Mosharaf Kabir Chowdhury and Raouf Boutaba. A Survey of Network Virtualization. *Computer Networks: The International Journal of Computer and Telecommunications Networking*, pages 862–876, April 2010.

[19] Hariharan Rahul, Mangesh Kasbekar, Ramesh Sitaraman, and Arthur Berger. Towards Realizing the Performance and Availability Benefits of a Global Overlay Network. Technical Report MIT-LCS-TR-1009, Massachusetts Institute of Technology, Computer Science and Artificial Intelligence Laboratory, 2005.

[20] R. Srinivasan. RPC: Remote Procedure Call Protocol Specification Version 2. RFC 1831, IETF, August 1995.

[21] Paul J. Leach and Dilip C. Naik. A Common Internet File System (CIFS/1.0) Protocol. Internet Draft, IETF, March 1997.

[22] WIDE Project. http://www.wide.ad.jp/, December 2012.

[23] WIDE Cloud Controller. http://wcc.wide.ad.jp/, December 2012.

[24] B. Callaghan, B. Pawlowski, and P. Staubach. NFS Version 3 Protocol Specification. RFC 1813, IETF, June 1995.

[25] J. Satran, K. Meth, C. Sapuntzakis, M. Chadalapaka, and E. Zeidner. Internet Small Computer Systems Interface (iSCSI). RFC 3720, IETF, April 2004.

[26] Peter Radkov, Li Yin, Pawan Goyal, Prasenjit Sarkar, and Prashant Shenoy. A Performance Comparison of NFS and iSCSI for IP-Networked Storage. *USENIX*, pages 101–114, 2004.

[27] W. Townsley, A. Valencia, A. Rubens, G. Pall, G. Zorn, and B. Palter. Layer Two Tunneling Protocol "L2TP". RFC 2661, IETF, August 1999.

[28] D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina. Generic Routing Encapsulation (GRE). RFC 2784, IETF, March 2000.

[29] M. Mahalingam, D. Dutt, K. Duda, P. Agarwal, L. Kreeger, T. Sridhar, M. Bursell, and C. Wright. VXLAN: A Framework for Overlaying Virtualized Layer 2 Networks over Layer 3 Networks. Internet Draft, IETF, August 2012.

[30] Luca Deri and Richard Andrews. N2N: A Layer Two Peer-to-Peer VPN. In *Proceedings of the 2nd international conference on Autonomous Infrastructure, Management and Security: Resilient Networks and Services*, AIMS '08, pages 53–64, 2008.

[31] Cisco Systems, Inc. `http://www.cisco.com/`, December 2012.

[32] VMware, Inc. `http://www.vmware.com/`, December 2012.

[33] R. Boivie, N. Feldman, Y. Imai, W. Livens, and D. Ooms. Explicit Multicast (Xcast) Concepts and Options. RFC 5058, IETF, November 2007.

[34] Yatin Chawathe. Scattercast: an adaptable broadcast distribution framework. *Multimedia Systems*, pages 104–118, July 2003.

[35] ntop. `http://www.ntop.com/`, December 2012.

[36] 河西朝雄. *C* **言語によりはじめてのアルゴリズム入門**, pages 337–343. 技術評論社, September 1994.

[37] The Linux Kernel Archives. `http://kernel.org/`, January 2013.

[38] KDDI Web Communications, Inc. `http://www.kddi-webcommunications.co.jp`, January 2013.

[39] CloudCore VPS Service. `http://www.cloudcore.jp/vps/`, January 2013.

[40] EditNet, Inc. `http://www.edit.ne.jp/`, January 2013.

[41] EditNet Flets Internet Connection Service. `http://www.editnet.ad.jp/services/flets.htm`, January 2013.

[42] Flets Next Internet Connection Service. `http://flets.com/next/`, January 2013.

[43] SAKURA Internet, Inc. `http://www.sakura.ad.jp`, January 2013.

[44] EditNet Operation Information. `http://intereddy.edit.ne.jp/disclosure-detail.php`, January 2013.

[45] Bonnie++: Disk and File System Performance Benchmark Utility. `http://www.coker.com.au/bonnie++/`, January 2013.

[46] Kernel-based Virtual Machine. `http://www.linux-kvm.org/page/Main_Page`, January 2013.

[47] Linux Advanced Routing & Traffic Control. http://lartc.org/, January 2013.

[48] Information Sciences Institute. TRANSMISSION CONTROL PROTOCOL. RFC 793, IETF, September 1981.