卒業論文 1997 年度 (平成 9 年度)

片方向通信路を含むネットワーク アーキテクチャにおける 動的な仮想リンク制御機構の設計と実装

慶應義塾大学 環境情報学部
西田 視磨

卒業論文要旨 1997 年度 (平成 9 年度)

片方向通信路を含むネットワークアーキテクチャにおける動的な仮想リンク制御機構の設計と実装

概要

本論文では、単一方向の通信路と従来の双方向通信路を組み合わせたネットワークアーキテクチャを使用するための動的な仮想リンク制御機構の設計と実装について述べる。

インターネットの急速な発展に伴いインターネットの利用者人口は増大の一途を辿っており、現在これら利用者への接続性に対する要求が高まっている。しかし、利用者に対する接続性を提供する既存の通信路であるアナログ電話回線や ISDN 回線は、帯域が狭い、接続手順が煩雑である等の点から、この接続性を提供するのに充分ではない。現在のインターネット上のトラフィックの特徴には、利用者から外部へのものは少なく、その反対方向は非常に多いという特徴がある。既存のインターネットにおける通信路である双方向の帯域が対称な通信路では、このトラフィックの特徴に適したネットワークを構築することが困難である。

片方向通信路は単一の方向へのデータ伝送を行う通信路である。片方向通信路を使用したネットワークは上述のトラフィックの特徴に適合し、低コストで広範囲・多数の利用者にネットワークの接続性を提供するものとして期待されている。しかし、インターネットにおける通信技術は通信路の双方向性を前提として設計されているため、片方向通信路と親和性が低い。したがって、現在のインターネットにおいて片方向通信路を使用する際には、多くの問題が発生する。

本論文では、片方向通信路をインターネット上の通信路として使用する際に発生する問題の解決を行った。特に、実用に際して既存の手法で問題となる大規模性に着目し、これを解決するため、片方向通信路を運用するための仮想リンク及びインターフェースを動的に制御する機構を設計し実装した。片方向通信路を含むテストベッド・ネットワークを構築し本手法の評価を行い、本手法が実用に充分な性能を持つことを検証した。これによって本論文で提案する手法が片方向通信路を含むネットワークにおいて有効であることを示した。

本論文で提案する手法により、片方向通信路と双方向通信路を組み合わせた、現在のインターネットにおけるトラフィックパターンに適合した有効なネットワークアーキテクチャが構築できた。

キーワード

インターネット　片方向通信路　仮想リンク　動的制御

Abstruct of Bachelor's Thesis Academic Year 1997

The Design and Implementation of Dynamic Configuration Mechanism for Virtual Links in Networks with Unidirectional Links

Abstruct

This paper discribes the design and implementation of dynamic configuration mechanism for virtual links and interfaces in networks with unidirectional links.
Nowadays Internet user has been increase accompanied by evloving Internet rapidly, requirement of connectivity for these users are building up. But conventional connectivity as analog telephone line or ISDN line is not enough to provide connectivity for many Internet users. Current Internet traffic pattern have hallmark, outgoing traffic is more and incoming one is few. Conventional links do not fit with this traffic pattern because these have symmetric bandwidth.
Unidirectinal link carries datagrams one way. Unidirectional link provides network connectivity for many internet users, widely and low costs. Unidirectional Link is expected as congruent link for nowadays traffic pattern of Internet. But design of communication technology in Internet is supposing bidirectional-capability of link. Unidirectional link has many problems at using in Internet.
This paper discribes the solution of ploblems using unidirectional link in Internet. Particularly, scalability is very important. For solving this problem, designed and implemented a mechanism of virtual links and interfaces dynamic configuring, construct testbed and evaluate this mechanism.
This mechanism made it come ture that the network archtecture feasible current traffic pattern of Internet with unidirectional and bidirectional links.

Keywords

Internet Unidirectional Link Virtual Link Dynamic Configuration

目 次

表 目 次

図 目 次

第 1 章

序論

1.1 研究の背景

現在、インターネットの爆発的な成長に伴って、利用者数は増加の一途をたどっている。インターネットが社会へ浸透すると共に、個々の利用者のインターネットへの接続性に対する要求も高まっている。個々の利用者とインターネットを接続する通信路には、ISDN 回線やアナログ電話回線を使用したものが一般的である。しかしこれらの通信路は帯域が狭く、運用の柔軟性に欠けるという欠点がある。しかし、高速の専用線接続はコストが高く、個人などでこれを持つのは難しい。これに対して近年、低コスト・広帯域の回線を一般家庭等に提供する通信路として、片方向通信路が登場した。これは、単一方向へのデータ伝送を行う機能を持つ通信路である。これらの代表的なものには、ケーブルテレビ網を利用したものや、衛星回線を使用したものなどが挙げられる。

片方向通信路をインターネット上の通信路として使用したネットワークアーキテクチャは、現在のインターネットにおけるトラフィックの特徴によく適合するものとして期待されておりその実現が対する要求が高まっている。現在のインターネットにおけるトラフィックには、利用者から外へ向かうトラフィックは少なく、逆に外から利用者に向かうトラフィックは多いという特徴がある。これは、大多数の利用者にとって情報獲得手段としてのインターネットの利用が一般化したためである。インターネットにおける情報提供サービス方式の一つである World Wide Web が非常に広範囲に浸透したことが、このようなトラフィックの特徴を生んだ最大の要因である。従来のインターネットは、双方向の通信を行う通信路で構成されてきた。しかし、双方向通信路による構成を前提とした従来のネットワークアーキテクチャでは、利用者人口の増加によって上述の特徴を持ち急激に増加するトラフィックに対して対応が困難である。

こうした要求から、広帯域単一方向の通信路を使用したネットワークの構築が研究されてきた。WIDE Project では、片方向通信路のうち特に静止軌道通信衛星を使用した衛星回線に着目し、WIDE Internet with Satellite Hermonization (WISH) という研究グループを構成してこの研究を行ってきた。衛星回線は、広帯域で均質の接続性を地理的に広い範囲に、効率的に提供できる通信路である。高品質なネットワークを

広域に且つ均質に提供する媒体として注目されている。衛星回線は、1 つの送信設備に対して非常に多数の受信設備を接続することが可能であり、受信設備は低コストで提供できる、と言う特徴がある。また、インターネットにおいて効率よく情報を配布するための技術として、IP Muticast がある。衛星回線は、その地理的普偏性と均質である点から、マルチキャスト通信と非常に親和性の高い通信路である。こういった特質から、効率よく情報を配布する媒体として期待されている。

これまで WISH では、従来のインターネットにおける通信技術と片方向通信路である衛星回線の親和性を高めるための研究を進めてきた。その成果として、衛星回線という伝送路をインターネットで使用するための問題解決と技術の開発が行われた。これによって、衛星回線をインターネット上の伝送路として使用することが可能になった。また、衛星回線を片方向の通信路として使用する際に発生する経路制御の問題を解決するための研究を行った。この成果として、トンネリング技術を用いて衛星回線における経路制御を動作させる機構を実現した。

本論文ではこれまでの研究を踏まえた上で、片方向通信路と双方向通信路を組み合わせた、大規模性と柔軟性のある効率のよいネットワークアーキテクチャの実現について論じる。片方向通信路をインターネット上で用いる際に発生する問題を明らかにし、従来の研究で解決されていなかった大規模性に起因する問題の研究を行った。片方向通信路を含むネットワークアーキテクチャにおいて大規模性を向上させるため、インターフェースおよび仮想リンクを動的に制御する機構を提案する。これらに必要な要件について論じ、設計ならびに実装を行う。また本論文で提案する手法が有効であることを示すため、片方向通信路と双方向通信路を組み合わせたテストベッドを構築し、本手法の評価を行について述べる。

1.2 本論文の構成

第 2章では、本論文で取り扱う片方向通信路について論ずる。第 3章では、関連研究について述べ、本論文との比較について述べる。第 4章では、片方向通信路をインターネットで使用する際に発生する問題の解決方法について述べる。第 5章では、本論文で提案する動的仮想リンク制御機構の設計について述べる。第 6章では、本論文で提案する機構の実装について述べる。第 7章では、本機構の評価について述べる。第 8章では、結論と今後の課題について述べる。

第 2 章

片方向通信路

本章では、本論文で取り上げる片方向通信路について述べる。

2.1 片方向通信路の定義

片方向通信路は、データリンク層の機能として単一の方向にのみデータを伝送する通信である。通信路上には複数のノードが同時に接続される。接続されるノードには、データリンク層の機能として送信機能を持つノード（以下「送信局」と呼ぶ）と、受信のみの機能を持つノード（以下「受信局」と呼ぶ）が存在する。片方向通信路は、ノードの通信路に対する接続の形態によって 2 つに分類される。一つは、通信路上に接続されるノードが 1 つの送信局と 1 つの受信局のみによって構成される場合であり、もう一つは通信路上接続されるノードが複数の送信局と複数の受信局によって構成される場合である。近年登場してきた衛星回線を使用した片方向通信路やケーブルテレビ網を利用した片方向通信路は、後者にあたる。本論文では、特に断りのない限り、片方向通信路は後者の形態のものを指して言う。

本論文では、複数の送信局と受信局によって構成される片方向通信路を、以下のように定義する。

片方向通信路上に接続され且つ受信機能を持つノードは、この通信路上を伝送される全てのデータを受信する。個々のノードのインターフェースはデータリンク層アドレスを持ち、受信時にデータリンク層アドレスに基づいて、データが自分宛であるか否かを判断する。

このトポロジーを図 2.1に示す。文中および図中の Feeder は送信局を、Receiver は受信局を指す。UDL(UniDirectional Link) は片方向通信路を指し、BDL (BiDirectional Link) は双方向通信路を指す。片方向通信路における送信局と受信局は、単独のノードであるか、或いは複数のノードによって構成されるネットワークである。

2.2 片方向通信路を含むネットワークアーキテクチャ

片方向通信路をインターネット上で使用する場合を考える。
送信局は、片方向通信路へのインターフェースの他に外部のネットワークとの通信のためのインターフェースを持つ。

受信局は、片方向通信路に対して送信ができないので、外部のネットワークとの通信のために片方向通信路以外に 1 つ以上の他のネットワークに接続されるインターフェースを持つ。この構成を図 2.2に示す。

既存の片方向通信路である片方向の衛星回線やケーブルテレビ網では、送信設備のコストは高く、受信設備のコストは低いという特徴がある。複数の送信局と受信局によって構成される片方向通信路は、その多くがこの特徴を持つと考えられる。このため 1 つの片方向通信路は、少数の送信局と多数の受信局によって構成されるのが一般的である。これを図 2.3に示す。

2.3 片方向通信路の使用上の問題点

本節では、片方向通信路をインターネット上で使用する際の問題点について述べる。片方向通信路は、前述のようにデータリンク層の機能として単一方向にしか情報を伝送できない媒体である。これを現在のインターネットにおける通信技術を用いて使用する場合、片方向通信路特有の問題が発生する。以下で各々の問題について論ずる。

2.3.1 経路制御

インターネットに於ける経路制御は、OSI7 階層参照モデルのうちのネットワーク層で提供する。現在のインターネットにおける経路制御は、通信路の双方向性を前提としているため、その通信路上で相互に経路情報を交換することによって成り立っている。

RIP[14] などの既存の経路制御における一般的な実装では、隣接するルータの存在と、そのリンクのコストを相互に報告しあい、複数の報告された経路情報から最適な経路を選択する。隣接ルータとは、同一のデータリンクに接続されたルータの集合を指す。一般的な内部経路制御の実装の多くでは、隣接ルータを発見するためのパケッ

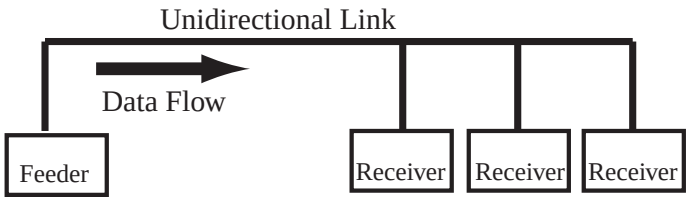

図 2.1: 片方向通信路ネットワーク・トポロジー図

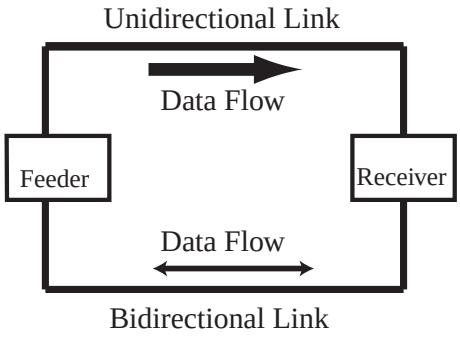

図 2.2: 片方向通信路と双方向通信路で構成されるネットワーク・トポロジー図

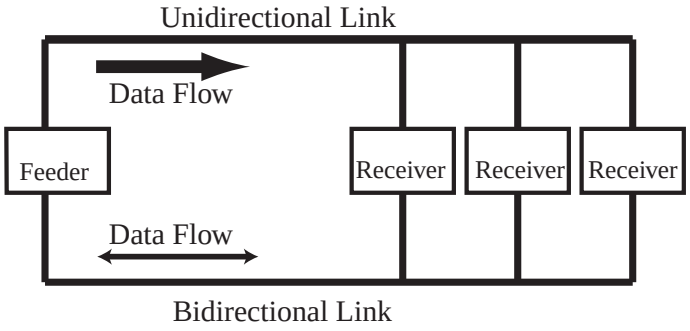

図 2.3: 一般的な片方向通信路を含むネットワーク・トポロジー図

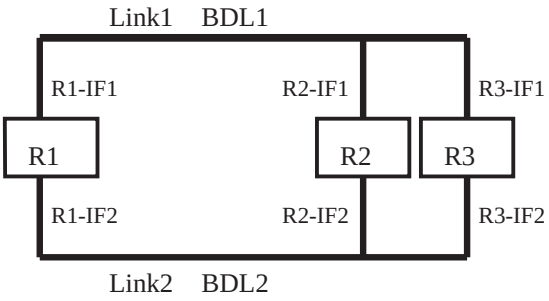

図 2.4: 双方向通信路で構成されたネットワーク

トや経路制御の情報伝達のためのパケットは、データリンク層ブロードキャストとして送信するように実装されている。

ここで、一般的な双方向通信路を使用したネットワークにおいて内部経路制御を適用した場合について述べる。図 2.4に示すネットワークを考える。

Rn はルータ、BDLn は双方向通信路を表す。BDL2 は、BDL1 よりもコストの高いリンクであるとする。Rn-IFn は、ルータ Rn の持つ BDLn へのインターフェースである。

この時、R2、R3 が R1 に対して経路を報告することによって、R1 は R2、R3 への経路を知ることができる。R1 は、R2、R3 へ経路を、BDL1、BDL2 の 2 つのインターフェースから受け取る。R1 は、R2、R3 宛のパケットの経路として、BDL1、BDL2 のリンクのうちから、コストの低い BDL1 をリンクを選択し、R1-IF1 よりパケットを送出する。

この経路制御を片方向通信路を含んだネットワークに適用すると、以下のような問題が発生する。上述のネットワークにおいて、link1 が UDL であった場合を考える。このネットワークを、図 2.5に示す。link1 は UDL であり、R1 は UDL の送信局、R2、R3 は UDL の受信局であるとする。BDL は、UDL よりコストの高いリンクであるとする。

片方向通信路では受信側は通信路にデータを送信することができない。したがって、R2、R3 から UDL に対してはデータを送信できない。UDL に対してデータを送信することができるのは、送信局である R1 だけである。この時の各 R が UDL 上の隣接ルータとして存在を検知できる R を考える。R1 は、R2、R3 からはデータを受信できないので、R1 は UDL の隣接ルータとして R2、R3 の存在を知ることができない。R2 は、R1 を隣接ルータとして検知することができるが、R3 は検知できない。同様の理由によって、R3 も R1 しか隣接ルータとして検知できない。

この時 R1 が受け取る経路情報は、R2、R3 が BDL を経由して送信するものだけとなる。この結果、R1 における R2、R3 へ経路は BDL を経由したものしか存在しない。したがって、R1 から R2、R3 へのパケットの経路は常に BDL が使用され、UDL は存在しコストも低いものであるにも拘わらず、使用されることがない。

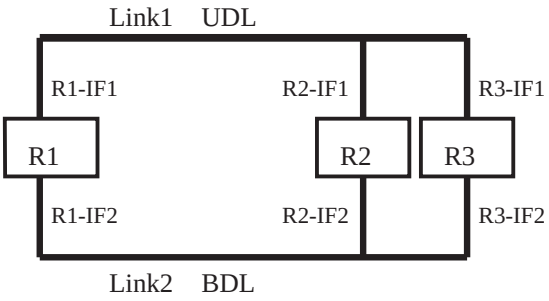

図 2.5: 双方向通信路と片方向通信路で構成されたネットワーク

このように、片方向通信路を含むしたネットワークでは、既存の経路制御は望ましい動作をしない。片方向通信路を含むネットワークを構築するには，この経路が使用されるような経路制御の仕組みが必要である。

2.3.2 データリンク層アドレス通知

2.1で論じたように、本論文で論じる片方向通信路はデータリンク層における同報機能を持つ。この時の片方向通信路における通信を考える。

多数のノードが同時に接続される通信路において、送信局から特定の 1 つの受信局にデータを送信する場合を考える。現在のインターネットにおいて一般的に使用されている多くの通信路では、送信局のデータリンク層で宛先のデータリンク層アドレスを知らなければならない。

Ethernet [5] の場合、ARP [6] と言う機構がこれを行っている。ARP は、送信を行いたいノードが、送信に先立って宛先のネットワーク層アドレスを持つノードに対してデータリンク層アドレスの問い合わせを行う。それに対して宛先ノードが返答する。その返答に基づいて、データリンク層でのパケット配送を行う。

しかし、片方向通信路では受信局から送信局へデータを送信することはできないので、受信局は片方向通信路を介してデータリンク層アドレスを送信局に通知することができない。従って送信局は送信相手となる受信局のデータリンク層アドレスを知ることができない。

これを解決するため、送信局が片方向通信路に属する受信局のデータリンク層アドレスを知る機構が必要となる。

2.3.3 通信路の状態検知

片方向通信路では、送信局は送信機能のみを持ち、受信局は受信機能のみを持つ。このため、送信局はもし送信先の受信局がデータリンク層において受信不能である場合、これを検知することは実装上困難である。同様に、受信局は送信局が送信不能で

ある場合、送信局と受信局が共に正常でありながら通信路の途上で何らかの障害が発生して不通となった場合も、これを検知する機構を実装することは難しい。既存の片方向通信路の実装では、この機構を提供したものはほとんどない。既存の片方向通信路では、こうした場合には、通信路は使用不能でもインターフェースは正常に動作しているように見える。

多くの経路制御の実装ではインターフェースの状態によってこの接続性を検知している。片方向通信路に接続された局が、片方向通信路以外のリンクに対してこの接続性を経路として報告している場合、通信路が不全でも使用可能経路として他に経路を報告してしまう。

片方向通信路をインターネット上の経路として使用する場合、片方向通信路が経路として使用不能になった際に、これを検出して経路に反映されるのは、経路制御プロトコルが片方向通信路を通過する経路を使用不能と判断する時となる。

この場合、実際の経路が経路制御の情報を反映するのに時間がかかるので、経路の収束に時間がかかる。

これを回避するために、片方向通信路の使用不能を検出し、それに応じてデータリンク層でインターフェースの接続・切断を制御する機構が必要である。

2.3.4 IP アドレスの動的な割当

現在インターネットにおいて IP アドレス資源の枯渇が大きな問題になっている。片方向通信路をインターネット上で使用する際には、従来の双方向通信路と同様に個々のノードに対して IP アドレスを付与する必要がある。片方向通信路を使用したネットワークアーキテクチャでは、非常に多数の受信局が接続されることが想定されており、これに必要な IP アドレス数も非常に大きなものとなるので、IP アドレス資源に対する圧迫が大きくなる。これを回避するために、片方向通信路を使用する時にだけ IP アドレスを使用するように個々の受信局に対して IP アドレスを動的に割り当てる機構が必要となる。

第 3 章

関連研究との比較

片方向通信路をインターネット上の通信路として使用するための技術は多く研究されている。本章ではこれら既存の技術の問題点と、本論文で提案する手法との比較について述べる。

3.1 経路制御プロトコルを改変する手法

2.3.1で述べた経路制御の問題を解決する手法として、経路制御を片方向通信路を含むネットワークアーキテクチャに適合するように改変する手法が提案されている。[1]

この手法は、片方向通信路を含んだネットワークの経路を処理可能なように経路制御プロトコルの設計を改変する。片方向通信路が多くの経路制御プロトコルにおいて問題となる理由は2.3.1で述べた。これを解決する手法として、経路制御プロトコルのパケットをネットワーク層のユニキャスト通信を使用して送信し、ネットワーク的に離れたルータに対して経路情報を配布できるようにする、という改変手法が提案されている。従来の設計を適用すると片方向通信路を経由して送信されるはずの経路制御パケットを、双方向通信路を使用して送信する。これによって、片方向通信路を含んだネットワークにおいて正しく経路制御が動作する。

しかしこの手法には、次に述べる問題点がある。

経路制御プロトコルを改変するこの手法では、インターネット上の全てのルータにおいて、改変された経路制御プロトコルが動作していなければならない。現在、インターネット上のルータの数は膨大なものである。これら全てのルータで動作している経路制御プロトコルを変更するには、膨大な費用と長い期間がかかる。またこの手法では、経路制御プロトコルごとに片方向通信路に適応するような設計をし直さなければならない。現在のインターネットでは経路制御プロトコルは複数用いられている。これらの全てについて個別の設計と実装を用意しなければならず、かかる労力が大きい。現在使用されている経路制御プロトコルについて片方向通信路に対応した設計を用意しても、新しい経路制御プロトコルが考案された時には、これに対しても個別に対応しなければならない。

この手法は、長期的な視野に立って、将来的に片方向通信路も視野に入れた全く新

しい経路制御プロトコルを設計するための前段階の研究としては意義がある。すでに運用されているインターネット上に、早期に片方向通信路の利用を実現するための技術ではない。

またこの手法は、片方向通信路を含むネットワークにおける経路制御についてのみを対象としており、2.3.2で述べたデータリンク層アドレス通知の問題、2.3.3で述べた通信路の状態検知の問題、2.3.4で述べた IP アドレスの動的な割当は、解決すべき問題の範疇に入っていない。

3.2 トンネリング技術を用いる手法

3.1で述べた手法に対して、短期的な視野での問題解決を行う手法として、片方向通信路の受信局から送信局への通信を、双方向通信路を使用したトンネリングによって行う手法 [2] が提案されている。

この手法では、2.2で述べたネットワークにおいて、受信局と送信局の間で双方向通信路を使用したトンネリングを行う。経路制御の設計では、片方向通信路の受信局から送信局へ、片方向通信路を介して送られなければならない経路制御パケットを、このトンネルを通して伝送することによって経路制御を正しく動作させる。

この手法は、3.1で述べた経路制御プロトコルを改変する手法と比較して、次の点で優れている。経路制御プロトコルを改変しないので、この手法を実装すれば、すぐにインターネット上で片方向通信路を使用可能である。また、既存の経路制御プロトコルがそのまま使用するので、在来のネットワークに片方向通信路を付加した場合、設備や設定に対する変更は片方向通信路の端点である送信局、受信局に対するもののみである。この点で、片方向通信路を使用したネットワークの構築にかかるコストが経路制御プロトコルを改変する手法と比較して少ない。

この手法は現在までにいくつかかが提案され、実装されているが、従来の手法や実装には次に述べる欠点がある。

個々の、受信局と送信局を結ぶトンネルの設定は静的である。2.2で述べたように、1つの片方向通信路に接続される受信局は多数であるのが一般的であるが、受信局の数が多数になった場合、静的な設定は大規模性に欠ける。また、片方向通信路が使用不能になった場合の対処は、人間が手動で行わなければならない。

この手法では、データリンク層アドレス通知、IP アドレスの自動割り当ては、解決すべき問題の範疇に入っていない。このため、これらの解決法については別の手法を用意する必要がある。

3.3 アドレス変換を用いる手法

2.2で述べたトポロジーのネットワークにおいて、受信局から外部のネットワークへの通信は双方向通信路を、外部のネットワークから受信局への通信は片方向通信路を使用するようにする技術として、アドレス変換 [7] [8] を使用した解決手法 [3] が提案さ

れている。

この手法は、受信局からのパケットの送信時にそのネットワーク層アドレスを書き換えることによって、そのパケットの通信における経路を選択するものである。インターネットの経路制御では、パケットの経路はそれに付与された Destination Address によって決定される。また、インターネットの通信では、受信したパケットの Source Address が返信するパケットの Destination Address として使用される。この点を利用し、受信局からのパケットの送信時にパケットの Source Address を片方向通信路のインターフェースのアドレスとすることによって、受信局から外部のネットワークへの通信には双方向通信路を、その返信には片方向通信路を使用する通信を実現する。

この手法は、片方向通信路のネットワークへの経路制御が設定されていることを前提として、パケットの経路をネットワーク層で制御するものである。片方向通信路を使用するか否かの選択は、受信局が行う。

この手法は前述の 2 つの手法と次の点で異なる。前述の手法は、パケットの経路を決定する経路制御に着目し、インターネットにおける通信全体が片方向通信路を使用できるようにするための技術である。これに対しアドレス変換を用いる手法では、経路がパケットに付与されたアドレスによって決定することに着目し、個々の通信において片方向通信路を使用する技術である。前述の 2 つの手法では、経路制御プロトコルが片方向通信路の使用を決定するのに対して、この手法では受信局が通信時にこの使用を決定する。

この手法の利点は、個々の通信に対して片方向通信路の使用を制御できるので、片方向通信路のトラフィックを柔軟に制御できる点である。またこの手法では、アドレスの変換によって通信路の使用を制御する。このため、複数の片方向通信路、双方向通信路が混在するネットワークにおいて、これらの使用を柔軟に制御できる。しかしこの手法は、次に述べる問題がある。パケットのネットワーク層アドレスを書き換えるため、セキュリティ技術との親和性に問題がある。特に IP Spoofing[9] 防止機構がこの手法を使用する上で障害となる可能性がある。

この手法では、片方向通信路への経路制御が設定されていることが前提となっており、2.3.1で述べた経路制御の問題は、この手法によって解決される問題の範囲外である。またこの手法は、ネットワーク層で片方向通信路の使用を制御するものであり、データリンク層の問題であるデータリンク層アドレス通知の問題、IP アドレスの動的な割当は、解決される問題の範囲外である。

3.4 既存の手法と本論文の手法との比較

本論文で提案する動的仮想リンク制御機構は、3.2 で述べたトンネリング技術を用いる手法を拡張したものである。したがって、本論文の手法では、既存の経路制御プロトコルに改変を加えることなく使用できる。

本論文で提案する手法では、トンネリング技術を用いる手法では静的に設定しているトンネルを、動的に制御する機構を提供する。これによって、データリンク層の制御の観点から見た片方向通信路の受信局の数に対する大規模性が大幅に向上する。本

機構ではデータリンク層アドレス通知機構、通信路の状態検知機構を同時に提供する。また、IP アドレスの自動割り当て機構を提供する。これによって、インターネット上の通信路として片方向通信路を用いる際の、受信局の数に対する大規模性が向上する。

本論文で提案する手法は、3.3で述べたアドレス変換による手法で問題となるセキュリティ技術との親和性の問題は発生しない。また本論文で提案する手法は、前述のトンネリング技術を用いる手法を拡張したものであるので、アドレス変換による手法とは 3.3で述べたように適用範囲が異なる。このため、両者を組み合わせて使用することによりこれらの利点をともに生かしたネットワークを構築することが可能である。本論文で提案する手法によって経路制御を動作させ、片方向通信路の使用をアドレス変換による手法によって制御することにより、片方向通信路を含むより柔軟性のあるネットワークアーキテクチャの実現が期待できる。

第 4 章

解決手法

本章では、片方向通信路をインターネット上で使用する際に問題となる点の解決方法について述べる。

4.1 片方向通信路における問題の解決

本節では、2.3で述べた問題を解決するための手法を述べる。

4.1.1 経路制御

図 2.5に示すネットワークにおいて、既存の経路制御プロトコルを使用し片方向通信路が使用されるためには、R2、R3 から R1 に対して、片方向通信路を経由して経路情報を伝達する必要がある。しかし片方向通信路は、受信側から送信側への送信は不可能である。

ここで、受信局と送信局が共に双方向通信路とも接続されている点を利用し、この問題を解決する。図 4.1に示すように、双方向通信路の接続性を使用してトンネリングを行うことにより、データリンク層で片方向通信路の受信局から送信局への復路を仮想的に設定する。受信局が複数のホストによってネットワークを構成している場合、受信局のネットワークで片方向通信路へのインターフェースと双方向通信路へのインターフェースは同一のホストが持ち、ルータとして機能するものとする。

送信局－受信局間の経路制御パケットは、送信局から受信局へは片方向通信路を、受信局から送信局へはトンネルを使用して交換される。
これによって構築できるリンクは、上位層から見て双方向の通信が可能である。この、片方向通信路のインターフェース、双方向通信路のインターフェース、トンネリングを利用した仮想的な復路の三つの要素によって実現されるリンクを仮想リンクと呼ぶ。仮想リンクによって実現される通信路において、送信局から受信局への方向を往路、受信局から送信局への方向を復路と呼ぶ。

4.1.2 仮想リンクに使用するトンネルと片方向通信路

4.1.1で述べた仮想リンクは、トンネリング技術を使用して実現する。トンネリング技術は、あるネットワーク層以上のプロトコルを伝送するために他の、あるいは同一のネットワーク層プロトコルをデータリンク層プロトコルとみなして利用する技術である。

片方向通信路の復路に使用するトンネルは、従来のトンネルとは異なる特徴をもつ。従来のトンネルは、上位層からはそれ自体が一つの双方向通信可能な通信路として捉えられる。しかし片方向通信路の復路として使用するトンネルは、受信局から送信局へのデータ伝送を行うが、その逆は行わない。このトンネルのデータ伝送の方向は、片方向通信路と反対である。このトンネルを片方向トンネルと呼ぶ。この片方向トンネルと片方向通信路を組み合わせて片方向通信路が上位層から見て双方向通信路と同様に送受信が可能なインターフェースとなるようにする。このモデルを、図 4.2に示す。

4.1.3 データリンク層アドレス通知

データリンク層アドレス通知の問題を解決する手法には、次の 2 つの方式が考えられる。一つは、送信側が送信時に受信者に対してデータリンク層アドレスを要求し、受信者の返答によってアドレスを知る方式である。この方式を On-Demand 方式と呼ぶ。

もう一つは受信側が片方向通信路の使用開始時に予めデータリンク層アドレスを送信側に通知し、送信側はそのデータを受信側の片方向通信路の使用終了まで保持しておく方式である。この方式を Pre-Regist 方式と呼ぶ。

これらの手法のうちどちらを採用すべきかは、片方向通信路の性質によって異なる。

衛星回線などの伝送遅延の比較的大きい通信路で On-Demand 方式のアドレス解決を行う場合、アドレスの要求を行ってからその返答が到着するまでの時間が長いため、通信に対して大きなオーバーヘッドになる。このような場合は、Pre-Regist 方式が有効である。

ケーブルテレビ網などの伝送遅延の比較的小さい通信路では、On-Demand 方式によってアドレス解決を行ってもそれに必要な時間は短い。送信側が全ての受信局のアドレスを保持するよりは、送信時に逐一アドレス解決を行う方が効率がよい。したがって、このような場合は On-Demand 方式が有効である。

4.1.4 通信路状態検知

片方向通信路の状態を検出するため、送信局は一定の間隔を置いて通信路にパケットを送出する。このパケットは、送信局が正しく動作を続けていることを表す短いパケットである。受信局は、このパケットを受信し続けている間、通信路が正常に動作していることを知ることができる。

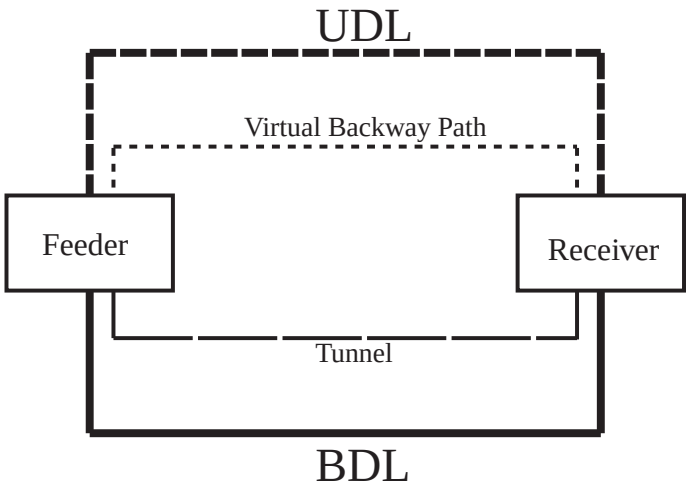

図 4.1: トンネルを使用した仮想リンクによって実現される通信路

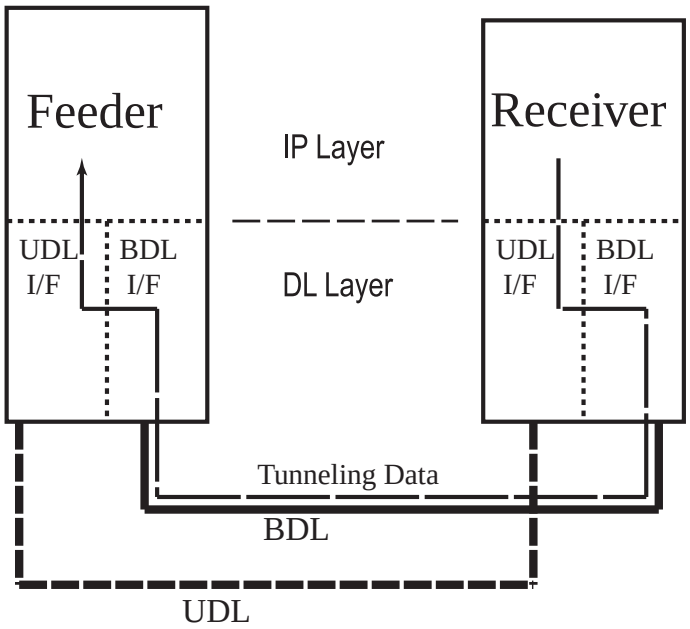

図 4.2: 片方向トンネルを使用した仮想リンクのモデル

4.2 動的仮想リンク制御・動的インターフェース制御

本節では、本論文で提案する動的仮想リンク制御機構、動的インターフェース制御機構と、その実現によって解決される問題について述べる。

4.2.1 仮想リンクの動的制御

4.1.2で述べたように本論文で提案する手法は、片方向通信路の持つ受信機能と双方向通信路の送信機能を用いたトンネルを組み合わせ、仮想的に双方向通信可能な片方向通信路のデータリンク層インターフェースを実現するものである。これに必要な仮想リンクを実現する片方向トンネルの設定には、受信局側で次に挙げる情報が必要である。

- トンネルの入口となる Receiver BDL I/F Address
- トンネルの出口となる Feeder BDL I/F Address

送信局の BDL I/F Address は不変ではないので、これが変更された場合は、トンネルの設定も変更しなければならない。2.2で述べたように、片方向通信路を含むネットワークのトポロジーでは、1 つの片方向通信路に接続される受信局の数は多数であるのが一般的である。トンネルの設定を手動で行うことを考えた場合、多数の個々の受信局において手動でトンネルを再設定するのは、大変な労力がかかり大規模性に欠ける。したがって、このトンネルの設定をネットワークの状態に併せて自動的に設定する機構が必要である。
本論文で提案するトンネルでは、トンネルの出口となる送信局に合わせてトンネルの入口となる受信局がトンネルを設定する。したがって、送信局と受信局の間で次のような機構を提供する。

送信局はトンネルを含む仮想リンクの状態を管理する。また送信局は、仮想リンクの状態についての情報を各受信局に通知する。受信局は仮想リンクとともに片方向通信路を使いたいときに、送信局に対して仮想リンクの設定許可を要求し、これを設定する。設定された仮想リンクに変更のあった場合は、送信局の通知に基づき自動的にこの制御を行う。
この機構によって、トンネリング技術を用いた片方向通信路の使用において、大規模性が大幅に向上する。

4.2.2 トンネリングの方式

トンネル技術を実現する方式には、現在までに多くの方法が提案されている。IP in IP Tunneling [11]、IP Encapslation within IP [12] [13]、General Routing Encapslation (GRE)[10] などがそれである。本節では、仮想リンクに用いるトンネリング・プロトコルについて論じる。

IP Encapslation within IP は、IP パケットを IP[4] でカプセル化して伝送するものであり、実装が容易である、カプセル化ヘッダが小さいためにトンネルによるオーバーヘッドが小さい、という特徴がある。GRE は、ルーティングのために設計された汎用のプロトコルであり、データリンク層アドレスも伝送できるが、実装が複雑なことや、トンネルによるオーバーヘッドが大きいという問題がある。

仮想リンクの実現にどのプロトコルを使用すべきかは、データリンク層アドレス通知の問題と関連することであり、片方向通信路の性質によって決まる。

すなわち、データリンク層アドレスの通知に On-Demand 方式を採用する場合、トンネリングによってデータリンク層アドレスも伝送できる GRE を用いるのが有効である。On-Demand 方式のデータリンク層アドレスの解決を行う場合、頻繁にデータリンク層アドレスの要求と通知が起こる。GRE を採用することによって On-Demand 式データリンク層アドレス通知に従来 Ethernet で用いられている ARP の機構をそのまま使用することができ、データリンク層アドレス通知機構を別個に実装するより実装が容易になる。また、頻繁にデータリンク層アドレスの解決を行うので、GRE のカプセル化ヘッダが通信に与えるオーバーヘッドと、既存のデータリンク層アドレス解決機構とは独立にデータリンク層アドレス通知機構を実装した場合のオーバーヘッドの差は、大きくならないと考えられる。

Pre-Regist 方式を採用する場合は、IP within IP を用いるのが有効である。Pre-Regist 方式の場合、データリンク層アドレス通知は 1 回しか起こらない。このため、データリンク層アドレス通知機構を既存のものと独立に実装し、トンネリングのプロトコルにデータリンク層アドレスを含まない IP within IP を使用することによって、全体の性能の向上が期待できる。

4.2.3 仮想リンクに使用する IP アドレスの自動設定

2.3.4で述べたように、片方向通信路を含むネットワーク・アーキテクチャが大規模性を保持するためには、IP アドレスの動的な割り当てが必要となる。これを解決する手法を次に述べる。

4.2.1で述べた機構において、仮想リンクの制御は受信局が片方向通信路を使いたいときに開始される。これを利用し、受信局が片方向通信路を使用する時のみ IP アドレスを割り当て、受信局がインターフェースを制御してこれを使用する機構を提供する。

受信局が片方向通信路の使用を開始する時、送信局は受信局に対して仮想リンクの使用を許可する。これと同時に、送信局はその受信局が片方向通信路インターフェースに割り当て可能な IP アドレスを通知する。受信局はこの通知に基づいてインターフェースに対して IP アドレスを割り当て、使用する。送信局は、各受信局に対して割り当てた IP アドレスにの状態を管理し、これらが重複しないように制御する。受信局が片方向通信路の使用を終了したときは送信局にこれを通知する。この時に、割り当てた IP アドレスの使用も停止し、この資源を解放する。

この機構によって、IP アドレス資源の観点から見た片方向通信路ネットワークの大規模性が向上する。

第 5 章

設計

本章では、片方向通信路をインターネット上で使用するための動的仮想リンク制御機構の設計について述べる。
本機構は、仮想リンク部と制御部に分かれる。

5.1 仮想リンク部の設計

仮想リンク部は、受信局の UDL インターフェースと送信局の UDL インターフェースを仮想的に接続するためのトンネリング機構を構成し、片方向通信路を、上位層に対して双方向通信可能なデータリンクとして提供する役目を持つ。トンネルの機能は、4.1.2で論じたように、受信側から送信側への単一方向のデータ伝送が可能であればよい。既存のトンネルの実装は双方向のトンネルである。仮想リンクにこれを使用することは可能であるが、送信側から受信側へ向けてのトンネルは使用されないため、この部分は不必要である。第 6章で述べる本論文の実装では、片方向通信路仮想リンク用のトンネルとして、単一方向に IP Encapslation within IP[12] [13] を使用したデータ伝送を行うものを使用した。このトンネリング機構は、既存の実装にはないものである。ここでは、このトンネリング機構を含め本論文の実装で採用した仮想リンク部の設計を述べる。

5.1.1 トンネル・モジュールの設計

トンネル・モジュールは、以下から構成される。

- 受信局側カプセル化モジュール
- 送信局側脱カプセル化モジュール

各モジュールは、受信局のデータリンク層・UDL インターフェース、送信局のデータリンク層・BDL インターフェースに実装される。これを図5.1に示す。図中、Encapslator はカプセル化モジュール、Decapslator は脱カプセル化モジュールを表す。

5.1.2 受信局側カプセル化モジュール

ネットワーク層から送信パケットがUDL インターフェースへ送られると、パケットはカプセル化モジュールに送られる。このパケットの Source Address は受信局の UDL I/F Address である。この時のカプセル化前のパケットの構成を、図 5.2に示す。

カプセル化モジュールは、このパケット全体を IP パケットのペイロードとしてカプセル化する。カプセル化ヘッダの Source Address には、トンネルの入口である Receiver BDL I/F Address、Destination Address には、トンネルの出口である Feeder BDL I/F Address を使用する。また、TTL を 255 に設定する。IP Encapslation within IP のプロトコル番号には、現在割り当てられていない 94 を採用した。よって Protocol フィールドに 94 を設定する。この時りのカプセル化後のパケットの構成を、図 5.3に示す。カプセル化されたパケットは、再びネットワーク層の送信ルーチンに渡される。

カプセル化を行う際に、トンネル入口のノードで繰り返しカプセル化されるループが発生するのを防ぐため、カプセル化前のパケットの Souurce Address、Destination Address およびプロトコルを検査する。カプセル化前パケットの Source Address とトンネルの入口のアドレス、Destination Address とトンネルの出口のアドレスがそれぞれ同一であり、Protocol フィールドが 94 である場合、このパケットはループしてカプセル化されているものとして廃棄する。

5.1.3 送信局側脱カプセル化モジュール

データリンク層 BDL インターフェースがパケットを受信した場合、脱カプセル化モジュールに送られる。
脱カプセル化モジュールは、パケットの Destination Address と Protocol フィールドを検査する。Destination Address が Feeder BDL I/F Address であり、Protocol フィールドが 94 である場合、パケットを脱カプセル化する。そうでない場合、通常の処理ルーチンに渡される。

脱カプセル化処理モジュールは、表 5.3に示すカプセル化ヘッダを取り除き、データリンク層 UDL インターフェースの受信ルーチンに渡される。

5.2 制御部の設計

制御部は、受信局側モジュールと送信局側モジュールに分かれる。受信局側モジュールは、送信局側モジュールとメッセージ交換を行い、仮想リンク部を制御する。送信局側モジュールは、受信局のデータリンク層アドレス、ネットワーク層アドレス、状態を管理し、受信局から送られるメッセージを処理する。

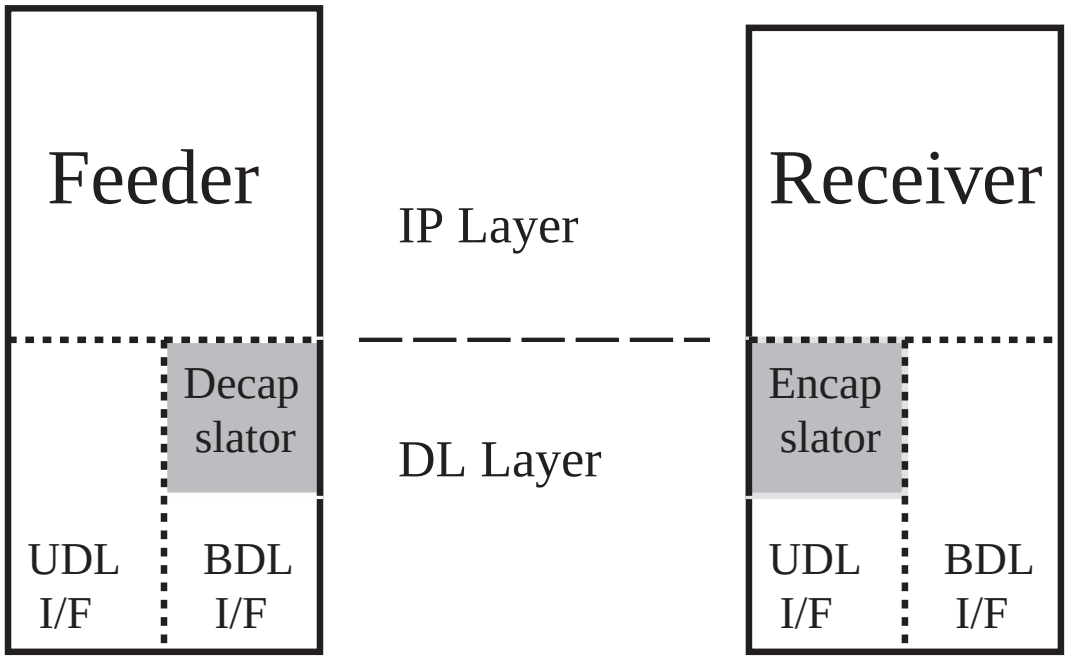

図 5.1: モジュール実装層

Destination Address	Somewhere Address
Source Address	Receiver UDL I/F Address
TTL	Original TTL
Protocotol	Original Protocol
Payload ...	

HEADER

PAYLOAD

図 5.2: カプセル化前のパケット構成

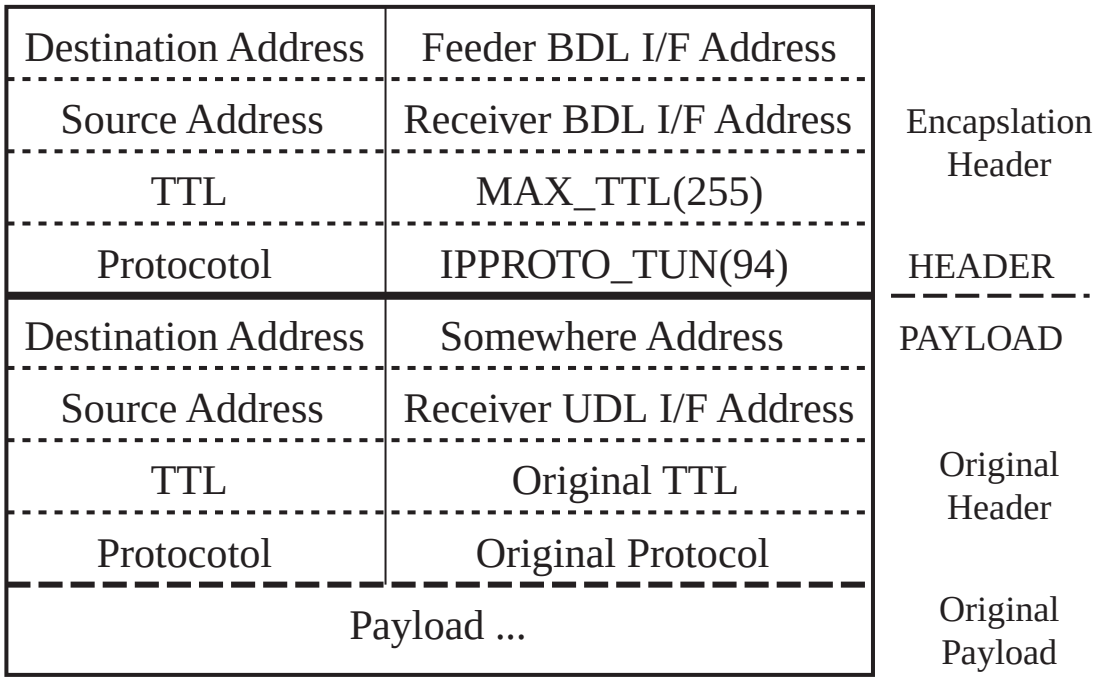

図 5.3: カプセル化後のパケット構成

5.2.1 本機構の基本的動作

本機構は、送信側と受信側が UDP を使用してメッセージ交換を行うことによって成立する。基本的な本機構の動作は、以下の通りである。

送信局は、UDL に対して定期的にメッセージを送信する。これを受信した受信局は、UDL が正常に動作中であることを知る。また、このメッセージから送信局の状態を知る。受信局が片方向通信路を使用したい場合、受信局から送出される内容に基づいて、BDL を経由して送信局に対して必要な情報の通知を行い、また必要な資源の割当を要求する。送信側は、受信側からの要求に基づいて、必要な情報の通知と資源の割当を行う。受信側が UDL の使用を終了した場合や、UDL が何らかの理由で使用不能になった場合は、受信局は割り当てられた資源の解放を送信局に申告する。これに基づいて送信局は割り当てた資源を解放し、受信局は UDL の使用を終了する。

5.2.2 メッセージ交換

動的仮想リンク制御機構は、以下の手順によりメッセージを交換する。メッセージ交換手順を図 5.4に示す。

1. Feeder は、UDL に対し一定間隔おきに HELLO メッセージを送出する。HELLO メッセージは Feeder の状態を各 Receiver に通知するとともに、UDL の KeepAlive の機能を果たすパケットとして利用される。

2. 動的仮想リンク制御機構を使用したい Receiver は、インターフェースを UDL に接続後、HELLO を受信するまで待つ。

3. HELLO を受信した Receiver は、HELLO 中にある Feeder の情報に基づき、BDL を経由して Feeder に対して Tunnel の設定要求と、必要な場合には IP アドレスの割当要求を行う。このメッセージを、REQUEST メッセージと呼ぶ。

4. Receiver の要求を受けた Feeder は、要求を受け入れるかどうかを判断し、その結果を BDL を経由して返答する。要求が受け入れられる場合には、Receiver が UDL が使用可能であることと使用可能な時間、必要ならば Receiver の UDL IP Address を返答に含む。要求が受け入れられない場合には、UDL 使用不可であることを返答に含む。このメッセージを、REPLY メッセージと呼ぶ。

5. 要求が受け入れられた場合、Receiver は インターフェース及び Tunnel を設定し、UDL を使用する。

6. 要求が受け入れられなかった場合、Receiver はランダムな時間待機した後、再び 2. の処理から繰り返す。

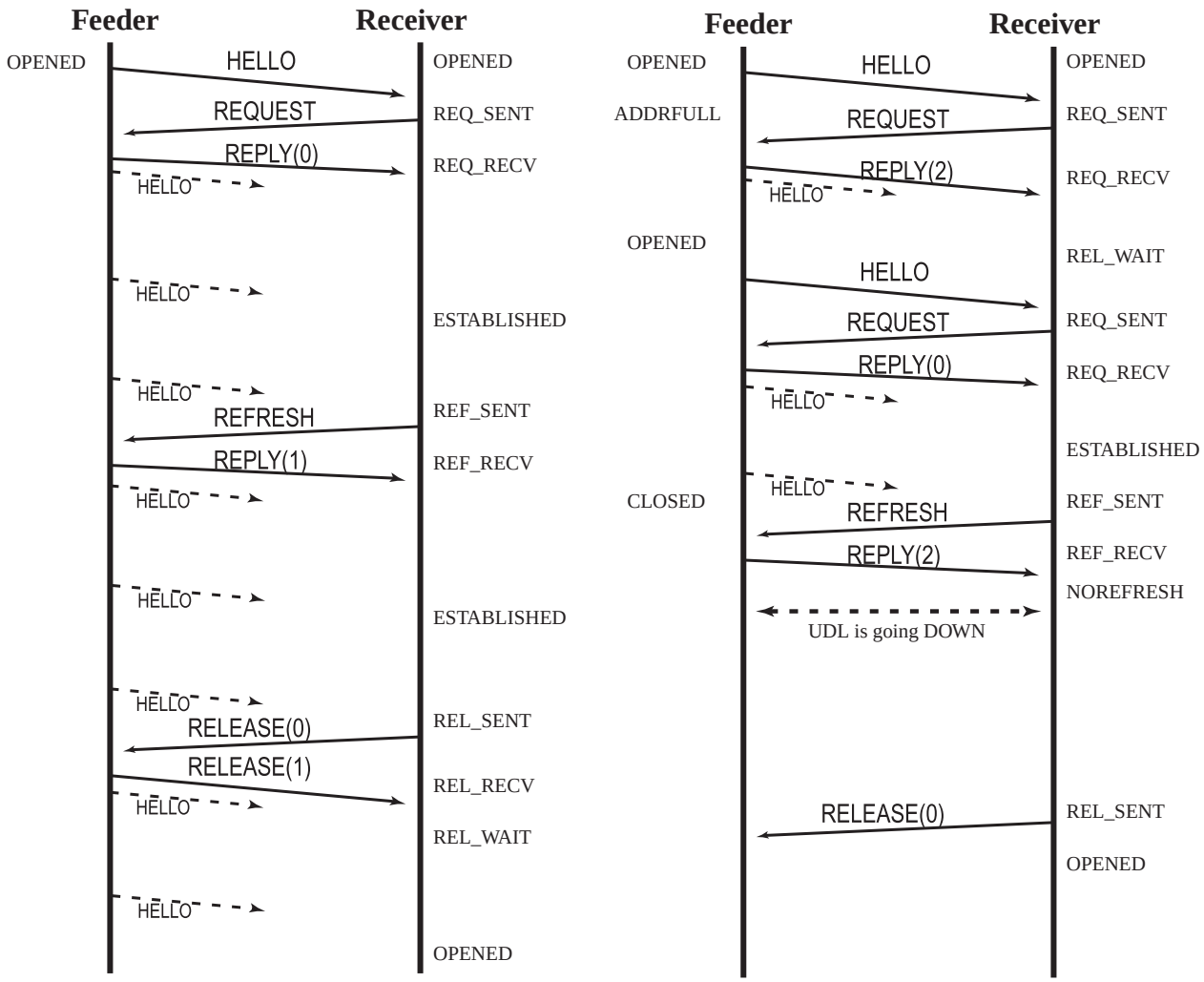

図 5.4: メッセージ交換図

7. Receiver が UDL の使用時間の期限が近づき且つ続けて UDL を使用したい場合は、BDL を経由して使用時間延長要求を Feeder に送信する。このメッセージを、REFRESH メッセージと呼ぶ。

8. Receiver が UDL の使用を終了したい場合は、BDL を経由して使用終了要求を Feeder に送信する。このメッセージを、RELEASE メッセージと呼ぶ。

9. Feeder が REFRESH メッセージを受信した場合、Receiver の使用延長を許可するか否かを判断し、許可する場合には REPLY メッセージを、許可しない場合は RELEASE メッセージを BDL を経由して送信する。

10. Feeder が RELEASE メッセージを受信した場合、該当の Receiver の割り当てていた資源を解放し、RELEASE メッセージを BDL を経由して送信する。

5.2.3 仮想リンクの使用許可と状態管理

受信局が仮想リンクを使用する場合、送信局に対して仮想リンクの使用を要求し、送信局がこれを割り当てることによってはじめて使用可能となる。送信局が仮想リンク使用要求に対して使用許可を与える時、要求が新規になされたものであった場合、新しく資源を割り当てる。使用許可をすでに与えている受信局から要求を受け取った場合、その使用許可がそのまま継続される。

また仮想リンクは、受信局がその仮想リンクを使用可能である時間(有効時間)を持つ。送信局は、仮想リンクの設定を許可してから、この時間をすぎて受信局が REFRESH メッセージによってこれを更新しなかった場合には、この仮想リンクは使用されなくなったものと判断してこのために割り当てていた資源を解放する。

受信局は、この有効時間を越えて更に仮想リンクを使用したい場合には、送信局に対して使用時間の延長を要求する。

5.2.4 受信局の状態管理

個々の受信局は、動的仮想リンク制御機構の動作を管理するため、図 5.5に示す状態を取る。受信局では、各メッセージの送信・受信がトリガになって、各状態間を遷移する。各状態の意味は、次に示す通りである。

- `CLOSED` (停止)
  制御部が動作を停止し、I/F を受信不能にしている状態を表す。

- `OPENED` (HELLO 受信可能)
  仮想リンク確立のため、I/F を受信可能にし HELLO メッセージの受信に待機している状態を表す。

- REQ_SENT　(REQUEST 送信)
  HELLO メッセージ受信後、REQUEST メッセージを送信する状態を表す。

- REQ_RECV　(REPLY 受信)
  REQUEST メッセージ送信後、送信局からの REPLY メッセージの受信に待機している状態を表す。

- ESTABLISHED (仮想リンク確立)
  仮想リンクが確立し、使用可能な状態であることを表す。

- REF_SENT　(REFRESH 送信)
  仮想リンクの使用有効期限が近づき、REFRESH メッセージを送信する状態を表す。

- REF_RECV　(REFRESH 受信)
  REFRESH メッセージ送信後、送信局からの REPLY メッセージの受信に待機している状態を表す。

- NOREFRESH　(仮想リンク更新不可)
  仮想リンク使用時間の延長ができず、有効期限経過後に仮想リンクの使用が不可能となる状態を表す。

- REL_SENT　(RELEASE 送信)
  仮想リンクの使用を終了し、RELEASE メッセージを送信する状態であることを表す。

- REL_RECV　(RELEASE 受信)
  RELEASE メッセージ受信後、送信局からの RELEASE メッセージの受信に待機している状態を表す。

- REL_WAIT　(待機)
  ランダムな時間待機している状態を表す。

5.2.5　送信局の状態管理

送信局は、動的仮想リンク制御機構の動作を管理するため、図 5.6に示す状態をとる。送信局では、片方向通信路の状態によって各状態間を遷移する。各状態の意味は、次に示すとおりである。

- OPENED　(UDL 使用可能)
  制御部が動作中であり、UDL が使用可能である状態を表す。一定間隔おきに HELLO メッセージを送出する。

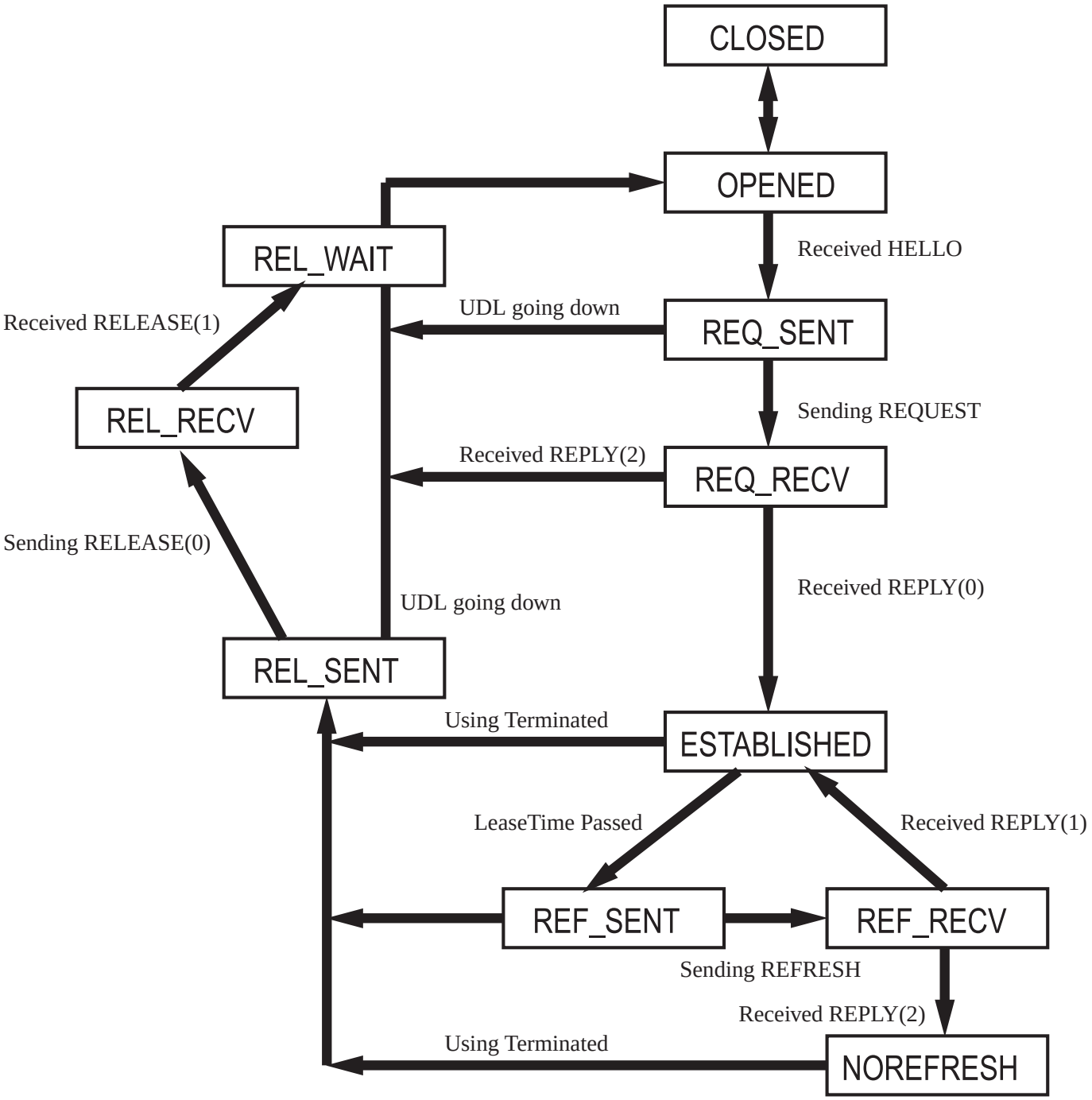

図 5.5: 受信局モジュール状態遷移図

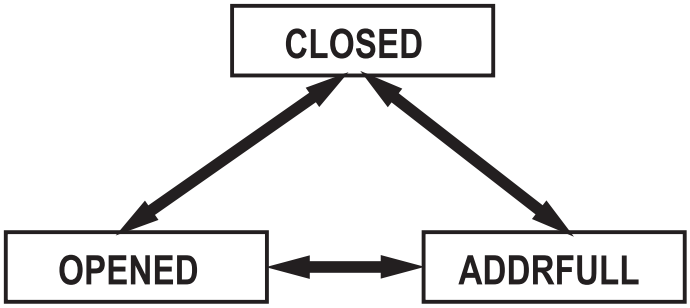

図 5.6: 送信局モジュール状態遷移図

- CLOSED　(UDL 使用不能)
  制御部が動作を停止し、UDL が使用不能である状態を表す。HELLO メッセージを送出しない。

- ADDRFULL (UDL 使用可能・IP アドレス割当不能)
  制御部が動作中であり、UDL が使用可能であるが、IP アドレスの動的割り当てを行うことが不可能な状態を表す。一定間隔おきに HELLO メッセージを送出する。

5.2.6　メッセージフォーマット

- HELLO メッセージ
  HELLO メッセージは図 5.7に挙げるフォーマットを持つ。

  HELLO メッセージは、送信局から UDL に対して、一定の間隔で常時送信される。このメッセージは UDL 上で同報される。データリンク層の宛先アドレス、ネットワーク層の宛先アドレスは、共にブロードキャストアドレスとなる。
  各フィールドの定義は、以下の通りである。

  - Operation Filed
    0. HELLO　　HELLO メッセージは、0 にセットされる。
  - Code Field
    0. ACCEPT　　Receiver がリンクの設定を要求してよいことを表す。
    1. DENY　　Receiver はリンクの設定を要求できないことを表す。
  - Interval
    HELLO メッセージの送出されている間隔を表す。単位は秒である。
  - Sequence Number
    HELLO メッセージのシーケンス番号。起動時にランダムな値から開始され、1 回 HELLO を送出するごとに 1 加算される。65535 の次は 0 となる。

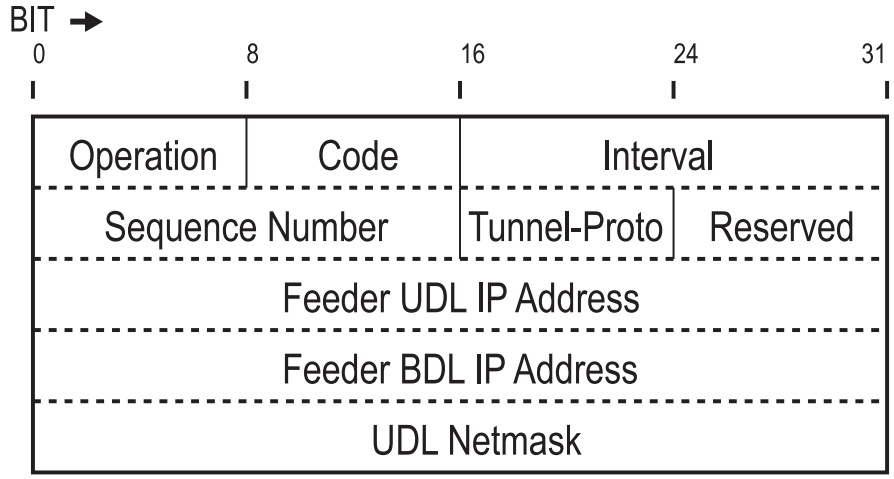

図 5.7: HELLO メッセージフォーマット

- Tunnel Proto
  Feeder が仮想リンクに使用しているトンネリングのプロトコルを表す。Receiver は仮想リンクのトンネルに、このプロトコルを使用しなければならない。
- Reserved
  常に 0 が設定される。
- Feeder UDL IP Addr
  Feeder の UDP I/F の IP Address がセットされる。このフィールドは、1 つの UDL に複数の Feeder が存在する場合の Feeder の ID として利用される。
- Feeder BDL IP Addr
  Feeder が仮想リンクのトンネル出口に使用している BDL I/F の IP Address がセットされる。Receiver は、仮想リンクのトンネル出口に、このアドレスを使用しなければならない。
- UDL Netmask
  UDL の ネットマスクを表す。

- REQUEST メッセージ
  REQUEST メッセージは図 5.8に挙げるフォーマットを持つ。

  REQUEST メッセージは、受信局が UDL を使用したい場合に、受信局から送信局に対して BDL を経由して送信される。

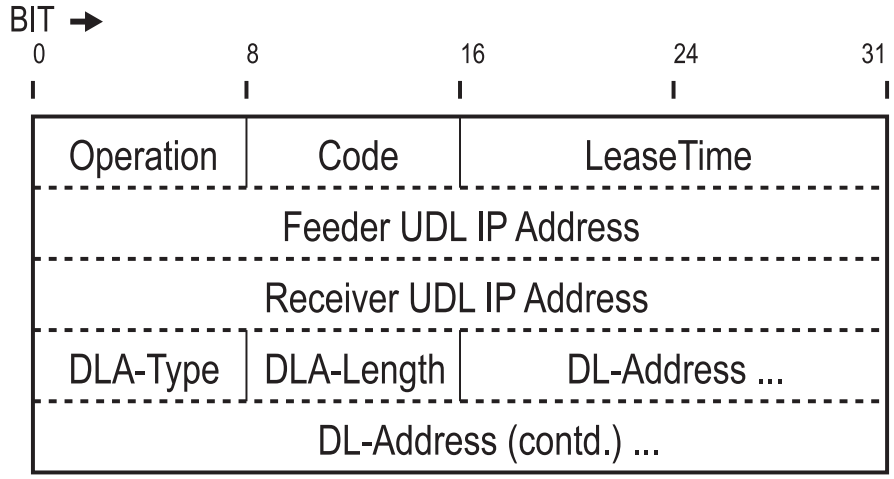

図 5.8: REQUEST メッセージフォーマット

- Operation Filed
  1. REQUEST　REQUEST メッセージは、1 にセットされる。
- Code Field
  REQUEST メッセージでは、常に 0 が設定される。
- Lease Time
  REQUEST メッセージでは、常に 0 が設定される。
- Feeder UDL IP Addr
  仮想リンクを設定する相手 Feeder の UDL I/F の IP Address がセットされる。
- Receiver UDL IP Addr
  Receiver の UDL I/F の IP Address がセットされる。IP Address の動的な割り当てを同時に受けたいときは、0 にセットされる。
- DL-Addr-Type
  Receiver UDL I/F のデータリンク層のタイプを表す。
  1. Ethernet
- DL-Addr-Length
  Receiver UDL I/F の UDL データリンク層アドレスの長さを表す。
- DL-Address
  Receiver UDL I/F の UDL データリンク層アドレスがセットされる。

- REPLY メッセージ
  REPLY メッセージは図 5.9に挙げるフォーマットを持つ。

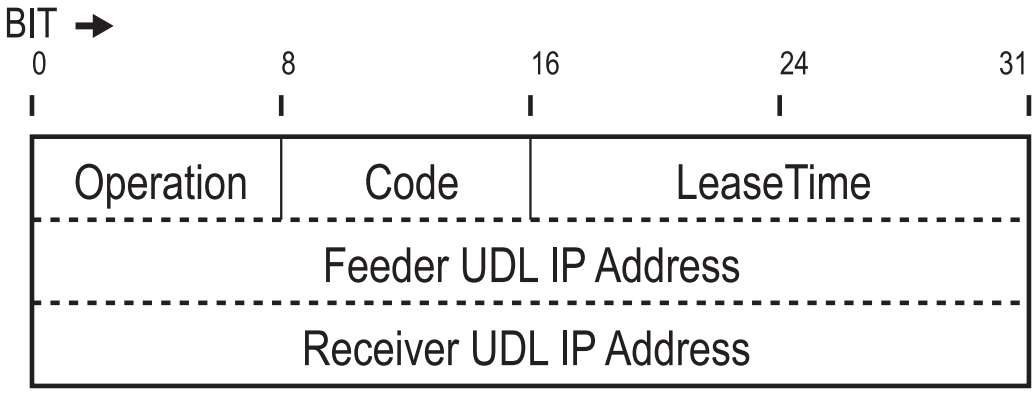

図 5.9: REPLY/REFRESH/RELEASE メッセージフォーマット

REPLY メッセージは、送信局が受信局からの REQUEST メッセージを処理した時に、その結果を通知するために送信される。このメッセージは、BDL を経由して受信局宛に送信される。

- Operation Filed
  2. REPLY　　REPLY メッセージは、2 にセットされる。
- Code Field
  0. REPLY_REQUEST_OK　Receiver の REQUEST 要求を受け入れたことを表す。
  1. REPLY_REFRESH_OK　Receiver の REFRESH 要求を受け入れたことを表す。
  2. REPLY_DENY　　Receiver の要求を拒否したことを表す。
- Lease Time
  仮想リンクを使用できる有効時間を表す。単位は秒である。
- Feeder UDL IP Addr
  Feeder の UDL I/F の IP Address がセットされる。
- Receiver UDL IP Addr
  Receiver の UDL I/F の IP Address がセットされる。IP Address の動的な割り当てを行う場合は、割り当てるアドレスにセットされる。

- REFRESH メッセージ

  RELEASE メッセージは図 5.9に挙げるフォーマットを持つ。
  REFRESH メッセージは、受信局が UDL の使用時間を延長して使用したい場合

に送信される。このメッセージは、BDL を経由して送信される。

- Operation Filed
  3．REFRESH　REFRESH メッセージは、3 にセットされる。
- Code Field
  0．RELEASE_REQUEST　Receiver が仮想リンクの使用を終了したことを表す。
  1．RELEASE_ACK　　Feeder が使用終了処理を行ったことを表す。
- Lease Time
  REFRESH メッセージでは、常に 0 がセットされる。
- Feeder UDL IP Addr
  Feeder の UDL I/F の IP Address がセットされる。
- Receiver UDL IP Addr
  Receiver の UDL I/F の IP Address がセットされる。

- RELEASE メッセージ
  RELEASE メッセージは図 5.9に挙げるフォーマットを持つ。
  RELEASE メッセージは、受信局が UDL の使用を終了又は中止する場合に送信される。このメッセージは、BDL を経由して送信される。

  - Operation Filed
    3．RELEASE　RELEASE メッセージは、3 にセットされる。
  - Code Field
    0．RELEASE_REQUEST　Receiver が仮想リンクの使用を終了したことを表す。
    1．RELEASE_ACK　　Feeder が使用終了処理を行ったことを表す。
  - Lease Time
    RELEASE メッセージでは、常に 0 がセットされる。
  - Feeder UDL IP Addr
    Feeder の UDL I/F の IP Address がセットされる。
  - Receiver UDL IP Addr
    Receiver の UDL I/F の IP Address がセットされる。IP Address の動的な割り当てを同時に受けたときは、以下の通りに設定される。
    1. Code = 0.RELEASE_REQUEST の時
       割り当てられたアドレスにセットされる。
    2. Code = 1. RELEASE_ACK の時
       常に 0 がセットされる。

5.3 障害検知・復旧

制御部は、片方向通信路における障害の検知と復旧を行う。片方向通信路における障害には、以下のものが挙げられる。

- 送信局の障害
- 受信局の障害
- 通信路の障害

本節では、本機構における障害の検知と復旧方法について述べる。

5.3.1 送信局の障害

送信局は、UDL に対して一定時間ごとに HELLO メッセージを送出している。受信局は、HELLO メッセージを正しく受信することによって、送信局が正常に動作していることを知ることができる。

ここで、送信局に障害が発生して送信不能になった場合を考える。送信局に障害が発生した場合、UDL に HELLO メッセージが送信されなくなる。受信局は、HELLO メッセージを一定時間受信しない場合、UDL が使用不能になったと判断し、送信局に UDL の使用終了を通告し、UDL の使用を終了する。UDL インターフェースを通信路から切断し、経路制御を BDL のみに切り替える。

これによって、送信局に障害が発生した場合に、自動的にネットワークから UDL を切り離すことができる。

送信局が障害から復旧した場合、再び HELLO メッセージを送出するので、受信局は送信局が正常に復したことを知ることができる。

送信局が極めて短時間に障害の発生と復旧をして仮想リンクの状態を管理するデータを消失した場合、受信局は HELLO メッセージのシークエンス番号が正常に増加していないことから、送信局がリブート等を行ったことを知ることができる。

この時は、受信局はその時点の仮想リンクを一旦リセットし、再び送信局に対して仮想リンクの設定を要求する。これによって、HELLO メッセージ送出間隔より短い時間内に発生した障害を検知することができる。

5.3.2 受信局の障害

送信局は、受信局に対して仮想リンクの使用を許可した場合、仮想リンクの有効時間を保持する。ここで、受信局に障害が発生し使用不能になった場合を考える。受信局が使用不能になった場合で受信局の BDL が動作している場合、受信局は UDL の使用終了を送信局に通告する。これによって送信局は受信局が使用不可能であることを知ることができる。

また、停電等により受信局が UDL の使用終了を通告せずに使用不可能になった場合

は、仮想リンク使用有効時間が経過すると自動的に受信局は UDL の使用を終了したと判断される。これによって、送信局は受信局の障害発生を検出することができる。この手法による受信局の障害検出は、仮想リンク有効有効時間の設定によって、長い時間がかかる可能性がある。しかし、一般的な経路制御の実装では、一定時間経路情報を受信しない経路については、経路から削除される。受信局の障害検出に経路制御の情報を使用することによってこの問題は回避できる。

受信局が復旧した場合、受信局は再び送信局に対して仮想リンク設定要求を送出し仮想リンクを設定するので、送信局は受信局の復旧を知ることができる。これは、仮想リンクの有効期限内に受信局に障害が発生し、復旧した場合も同じである。

5.3.3 通信路の障害

送信局、受信局の双方が正常に動作しているにも拘わらず、通信路が物理的な障害などで使用不能になった場合を考える。このとき、送信側からは受信局が使用不能になったように見え、受信局からは送信局が使用不能になったように見える。したがって、通信路の使用不能が一定時間以上続いた場合、送信局、受信局双方で、自動的に UDL を使用しない状態になる。

通信路の障害が復旧した場合、受信局が再び仮想リンク設定要求を行うので、自動的に仮想リンクは復旧する。送信局の HELLO メッセージの送出間隔より短い時間内に通信路に障害が発生し復旧した場合には、設定された仮想リンクはそのまま保持され、通信が持続される。

第 6 章

実装

本章では、第 5章で述べた動的仮想リンク制御機構の実装にについて述べる。

6.1 実装環境

実装は、FreeBSD 2.2.1-RELEASE 上で行った。
双方向通信路には、Ethernet を使用した。片方向通信路は、Ethernet のドライバを改造することによって実現した。

6.2 仮想リンク部の実装

仮想リンク部は、Unix カーネルのデータリンク層ルーチンを改造することによって実装した。

6.2.1 インターフェース情報の保持と制御

カーネル内でインターフェースの情報を保持している ifnet 構造体と、このデータの操作を行うための ifreq 構造体が、トンネルの情報を保持するように改造した。この ifnet 構造体、ifreq 構造体は、従来のものが持つ情報の他に、インターフェースが受信局であるか送信局であるか、BDL であるか UDL であるかの区別、トンネルの入口アドレス、トンネルの出口アドレスを持つ。これを、表 6.1、表 6.2、表 6.3に示す。

インターフェースが受信局の UDL I/F である場合、if_status は RCV_UDL に設定され、カプセル化を行う時に必要なトンネルの入口アドレス、出口アドレスが inpoint_addr, outpoint_addr に設定される。インターフェースが送信局の BDL I/F であり、トンネルの出口となる場合、if_status は FDR_BDL に設定され、inpoint_addr、outpoint_addr は 0 に設定される。

表 6.1: ifnet 構造体

```
struct ifnet {
        : (snip)
    u_char  node_status;
    u_long  inpoint_addr;
    u_long  outpoint_addr;
        : (snip)
};
```

表 6.2: ifreq 構造体

```
struct ifreq {
        : (snip)
    u_char  node_status;
    u_long  inpoint_addr;
    u_long  outpoint_addr;
        : (snip)
};
```

表 6.3: node_status 変数のステータス値

```
#define NORMAL  0x00 /* 通常の I/F */
#define RCV_UDL 0x01 /* 受信局 UDL I/F */
#define FDR_UDL 0x02 /* 送信局 UDL I/F */
#define FDR_BDL 0x04 /* 送信局のトンネル出口となる BDL I/F */
```

表 6.4: 追加した ioctl 値

```
    :(snip)
#define SIOCSUDL         _IOW('i', 55, struct ifreq)    /* set UDL if on */
#define SIOCGUDL        _IOWR('i', 56, struct ifreq)    /* get UDL if stat */
    :(snip)
```

前述の ifnet 構造体のトンネル情報の読み出し、書き込みに ioctl ルーチンを使用した。このため、sockio.h 中に UDL 制御用のメッセージを追加した。これを、表 6.4 に示す。また、if.c 中の関数 ifioctl() を改造した。

6.2.2 カプセル化・脱カプセル化ルーチン

本実装では片方向通信路、双方向通信路に Ethernet を用いたので、if_ethersubr.c 中の関数 ether_output にカプセル化モジュールを、ether_input に脱カプセル化モジュールを実装した。

インターフェースがパケットを送信しようとする時、ether_output ルーチンが呼び出される。ここで、インターフェースが受信局 UDL インターフェースであった場合、パケットはカプセル化ルーチンに渡される。カプセル化ルーチンは、5.1.2で述べたカプセル化処理を行う。その後、ip_input ルーチンを呼び出し処理したパケットを渡す。

インターフェースがパケットを受信した時、ether_input ルーチンが呼び出される。ここで、インターフェースがトンネル出口に設定された送信局 BDL インターフェースであった場合、パケットは脱カプセル化ルーチンに渡される。脱カプセル化ルーチンは、5.1.3で述べた脱カプセル化処理を行う。その後、ether_input ルーチンを呼び出しこの処理したパケットを渡す。

6.2.3 UDL エミュレーションルーチン

本実装では片方向通信路に Ethernet を用いたので、Ethernet を使用して片方向通信路をエミュレートするルーチンを if_ethersubr.c 中に実装した。

インターフェースが前述 6.2.1で述べた FDR_UDL に設定されていた場合、このインターフェースは送信専用インターフェースとして動作する。このインターフェースがパケットを受信した場合、そのパケットは全て破棄される。このインターフェースがパケットを送信する場合、データリンク層アドレス解決を行う際に arpresolv ルーチンを呼び出さず、後述する送信側モジュールの仮想リンク情報テーブルを検索し、送信相手先のデータリンク層アドレスを得る。

表 6.5: 仮想リンク情報テーブル

IP Address	Status	DL Address	Lease Time Left
:	:	:	:

6.3 制御部の実装

制御部は、ユーザプロセスとして実行されるプログラムとして実装した。

6.3.1 送信局側モジュールの実装

5.2で述べた設計に基づき、送信局側モジュールを実装した。送信局側モジュールは、次に挙げる項目を設定できる。これらを設定する設定ファイルの例を表6.6、表6.7に示す。

- UDL として使用するインターフェース (udl-if)
- トンネルの出口として使用するインターフェース (bdl-if)
- HELLO メッセージの送出間隔 (interval)
- 動的に割り当てたトンネル、IP アドレスの使用時間 (leasetime)
- 動的に割り当てる IP アドレス

送信局側モジュールは、仮想リンク情報テーブルを保持する。この構造を図6.5に示す。各エントリの意味を次に述べる。

- IP Address
  受信局 UDL IP Address
- Status
  IP Address の状態を表す。
  `0. OPENED` 動的割り当て可能
  `1. ASSIGNED` 割り当て済み
  `2. STATIC` 静的に割り当てた IP アドレス
- DL Address
  受信局 UDL IP Address に対応するデータリンク層アドレス
- LeaseTime Left
  仮想リンク使用有効時間の残り（秒）

表 6.6: 送信モジュール設定ファイル例

```
# DTCP Server ConfigFile SAMPLE

# Interface Configration
udl_if=ed0
bdl_if=de0

# DTCP Configration
interval=10
addr_db=dtcp_addr.pool
leasetime=1800
```

表 6.7: 送信モジュール IP アドレスプール設定ファイル例

```
# DTCP AddressPool ConfigFile Sample

# IP Address List for Client Assigning
ipaddr=172.16.1.2
ipaddr=172.16.1.3
ipaddr=172.16.1.4
ipaddr=172.16.1.5
```

このテーブルのエントリには、REQUEST メッセージから得た受信局の情報が保持される。

6.3.2 受信局側モジュールの実装

5.2で述べた設計に基づき、受信局側モジュールを実装した。受信局側モジュールは、次に挙げる項目を設定できる。これらを設定する設定ファイルの例を表6.8に示す。

- UDL として使用するインターフェース (udl-if)
- トンネルの入口として使用するインターフェース (bdl-if)
- HELLO 受信のために一時的に設定される IP アドレス (pseudo-ifaddr)
- HELLO 受信のために一時的に設定されるネットマスク (pseudo-netmask)
- トンネル、割当 IP アドレスの更新を開始する時間 (ref-border)
- メッセージを受信できなかったときのタイムアウト (ta-timeout)

表 6.8: 受信側モジュール設定ファイル例

```
# DTCP Client ConfigFile SAMPLE

# Interface Configuration
udl_if=ed0
bdl_if=ed1
pseudo-ifaddr=0.0.0.0
pseudo-netmask=255.255.255.255

# DTCP Configuration
ref_border=10
ta_timeout=30
udl_timeout=30
seq_threshold=100
```

- HELLO を受信しないとき UDL が使用不能と判断する時間 (udl-timeout)
- 送信局がリセットされたと判断する HELLO シークエンスの幅 (seq-threshold)

第 7 章

評価

本章では、第 6章で述べた動的仮想リンク制御機構の評価にについて述べる。

7.1 評価環境

本機構の評価のため、図 7.1に示すネットワークを構築した。このネットワークは、片方向通信路と双方向通信路をシミュレートする実験環境である。各ホストの OS には、FreeBSD 2.2.1-RELEASE を用いた。経路制御デーモンには、gated Revision3.5 Beta3 を用いた。各ホストの役割および各ネットワークの役割を、表 7.1, 表 7.2に示す。

7.2 経路制御の動作の評価

本機構を実装した片方向通信路を含むネットワーク上で、経路制御機構が正しく動作することを検証する実験を行った。経路制御プロトコルには、RIP [14] を使用した。
評価項目は、次に挙げる 2 つである。

表 7.1: 実験環境ネットワーク ホスト役割表

ホスト名	役割
aquarius	片方向通信路 送信局 (Feeder)
ulysses	片方向通信路 受信局 (1) (Receiver1)
valkyrie	片方向通信路 受信局 (2) (Receiver2)
zodiac	双方向通信路ネットワーク上のルータ (Router)
icarus	受信局ネットワーク内部のホスト (1) (Host1)
odysseus	受信局ネットワーク内部のホスト (2) (Host2)
orpheus	DNS サーバ (NameServer)

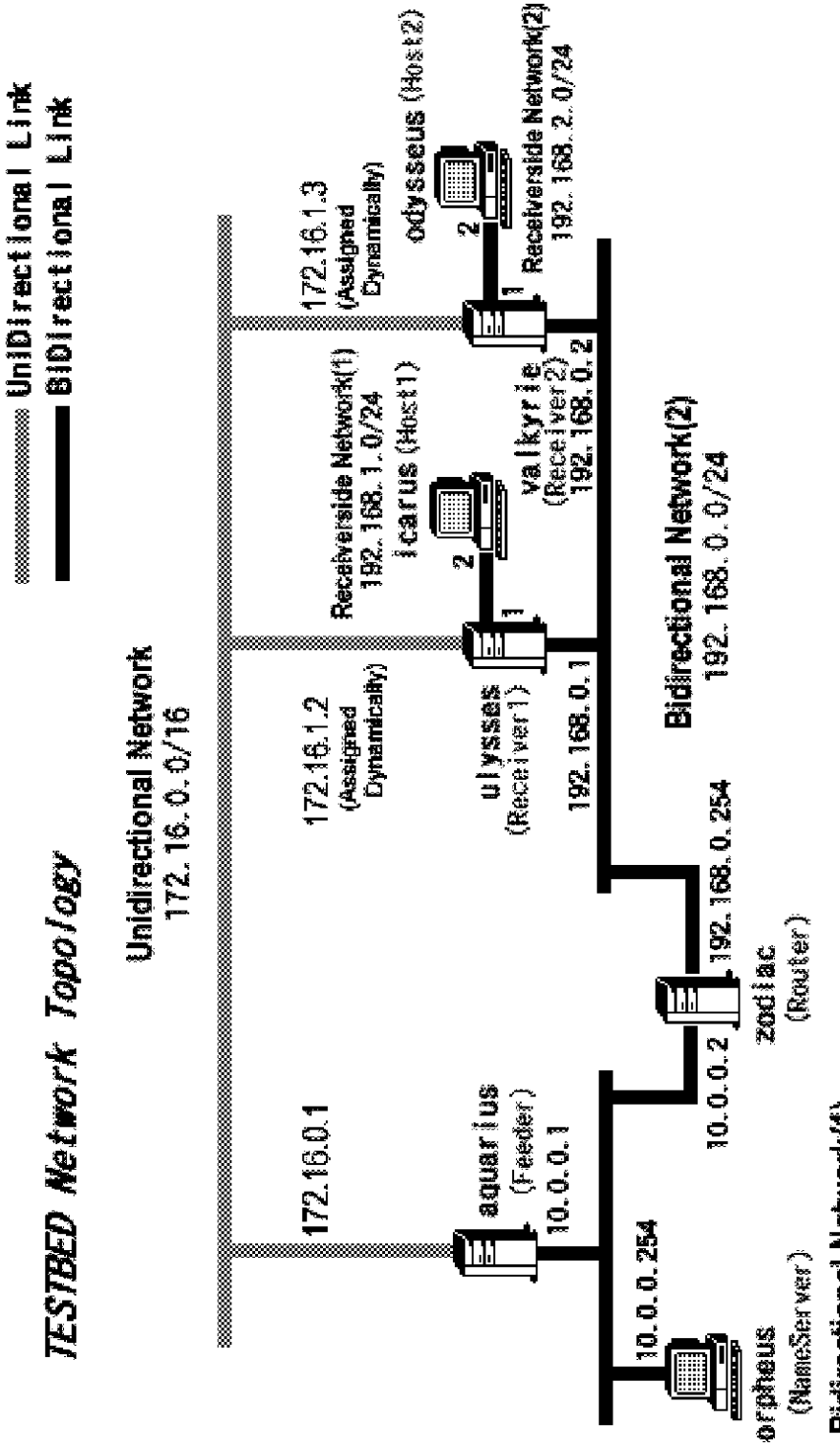

図 7.1: テストベッドネットワーク・トポロジー図

表 7.2: 実験環境ネットワーク ネットワーク役割表

ネットワークアドレス	役割
172.16.0.0/16	片方向通信路
10.0.0.0/16	双方向通信路 (1)
192.168.0.0/24	双方向通信路 (2)
192.168.1.0/24	受信局 (1) の内部ネットワーク
192.168.2.0/24	受信局 (2) の内部ネットワーク

- 片方向通信路が正常である場合、受信側ネットワーク上へのパケットの送信において片方向通信路が最適の経路となるばあい、これが使用される。
- 片方向通信路が不通となった場合、片方向通信路が経路として使用されない。

7.2.1 経路の設定

図 7.1で示すネットワークにおいて、以下のように経路を設定した。

- 送信局 aruarius において、UDL と BDL は等しく metric 1
- 受信局 ulysses、valkyrie において、BDL は metric 1、UDL は metric 2
- 双方向通信路上のルータ zodiac において、双方の BDL とも metric 1

送信局 aquarius は、片方向通信路に対して送信可能であるので、この経路に対するコストを metric 1 とした。受信局 ulysses、valkyrie は、片方向通信路に対する送信時はトンネルを使用するのでなるべくこのトンネルの使用をさけるようにするため、metric を双方向通信路より大きいコストとなる metric 2 とした。片方向通信路のネットワークからみて外部のネットワーク上のルータとなる zodiac は、2 つの経路のコストを等しい metric 1 とした。

7.2.2 片方向通信路が正常である場合の経路

片方向通信路が正常に動作している状態で、経路制御の動作を評価した。経路制御の状態が安定した時の送信局 aquarius, 受信局 ulysses(172.16.1.2) における経路表を、表 7.3, 表 7.4に示す。

表からわかるように、送信局 aquarius の経路表では、片方向通信路の受信局への経路に、片方向通信路のネットワークである 172.16/16 が設定されている。また、受信局 aquarius の経路表では、送信局や外部のネットワークへの経路に、片方向通信路以外のネットワークが設定されている。

表 7.3: 片方向通信路正常時の送信局 aquarius の経路表

```
Routing tables

Internet:
Destination        Gateway            Flags     Refs     Use     Netif Expire
10/16              link#1             UC           0       0
10.0.0.2           52:54:4c:4:30:4a   UHLW         1       2       de0   1182
10.0.0.254         0:47:43:11:80:1    UHLW         0     158       de0    251
127                127.0.0.1          URc          0       0       lo0
127.0.0.1          127.0.0.1          UH           0       0       lo0
172.16             link#2             UC           0       0
172.16.1.2         0:0:e8:d5:86:61    UHLW         1       0       ed0   1132
172.16.1.3         0:80:ad:16:86:38   UHLW         1       0       ed0   1087
172.16.255.255     ff:ff:ff:ff:ff:ff  UHLWb        0       3       ed0
192.168            10.0.0.2           UGc          1      12       de0
192.168.1          172.16.1.2         UGc          0      10       ed0
192.168.2          172.16.1.3         UGc          0       5       ed0
224.0.0.9          127.0.0.1          UH           1       1       lo0
```

表 7.4: 片方向通信路正常時の受信局 ulysses の経路表

```
Routing tables

Internet:
Destination        Gateway            Flags     Refs     Use     Netif Expire
10                 192.168.0.254      UGc        620     626       vx0
127                127.0.0.1          URc          0       0       lo0
127.0.0.1          127.0.0.1          UH           0       0       lo0
172.16             link#2             UC           0       0
192.168            link#1             UC           0       0
192.168.0.2        52:54:4c:4:30:64   UHLW         1       1       vx0   1023
192.168.0.254      52:54:4c:2:22:67   UHLW         2       2       vx0   1032
192.168.1          link#3             UC           0       0
192.168.1.2        0:40:b4:80:34:3f   UHLW         1      16       ed1   1038
192.168.2          192.168.0.2        UGc          0       0       vx0
224.0.0.9          127.0.0.1          UH           1       5       lo0
```

表 7.5: 片方向通信路正常時の aquarius から icarus への traceroute

```
aquarius % traceroute icarus
traceroute to icaurs.kirc.wide.ad.jp(192.168.1.2): 1-30 hops, 38 byte packets
 1  172.16.1.2 (172.16.1.2)  1.510 ms  1.068 ms  1.022 ms
 2  icarus (192.168.1.2)  1.929 ms  1.875 ms  1.776 ms
```

表 7.6: 片方向通信路正常時の icarus から aquarius への traceroute

```
icarus % traceroute aquarius
traceroute to aquarius.kirc.wide.ad.jp(10.0.0.1): 1-30 hops, 38 byte packets
 1  ulysses (192.168.1.1)  1.935 ms  1.393 ms  1.857 ms
 2  zodiac (192.168.0.254)  2.112 ms  1.920 ms  1.901 ms
 3  aquarius (10.0.0.1)  2.330 ms  2.209 ms  2.097 ms
```

この時のネットワークでのパケットの経路は、次の通りである。

送信局やそれに近いネットワーク上のホストから受信局へのパケットの経路は、片方向通信路を経由する。この時 aquarius 上で icarus に対して traceroute を行った結果を、表 7.5 に示す。また、受信局から外部のネットワークへのパケットの経路は、双方向通信路を経由する。この時 icarus 上で aquarius に対して traceroute を行った結果を表 7.6に示す。

7.2.3 片方向通信路が不通である場合の経路

前述 7.2.2から、片方向通信路が不通となる状態を作り、経路制御の動作を評価した。片方向通信路が不通となる状態は、片方向通信路が物理的に接続されている通信路途上の HUB を機能しないようにすることによって実現した。経路制御の状態が安定した時の送信局 aquarius, 受信局 ulysses(172.16.1.2) における経路表を、表 7.7, 表 7.8 に示す。

表からわかるように、送信局 aquarius の経路表では、片方向通信路の受信局への経路に双方向通信路のネットワークである 10.0/16 が指定されている。また、受信局 ulysses の経路表では、動的仮想リンク制御機構によって UDL I/F が通信路から切断され、この経路がなくなっており、他のネットワークとの通信で全て双方向通信路を経由するように設定されている。

この時のネットワークでのパケットの経路は、次の通りである。

送信局やそれに近いネットワーク上のホストから受信局へのパケットの、受信局から外部のネットワークへのパケットの経路は、ともに双方向通信路を経由する。この時 aquarius 上で icarus に対して traceroute を行った結果を表 7.9に、icarus 上で aquarius

表 7.7: 片方向通信路不通時の送信局 aquarius の経路表

```
Routing tables

Internet:
Destination        Gateway            Flags      Refs      Use      Netif Expire
10/16              link#1             UC            0        0
10.0.0.2           52:54:4c:4:30:4a   UHLW          8      718        de0    913
10.0.0.254         0:47:43:11:80:1    UHLW          0       27        de0   1030
127                127.0.0.1          URc           0        0        lo0
127.0.0.1          127.0.0.1          UH            0        0        lo0
172.16             link#2             UC            0        0
172.16.1.2         0:0:e8:d5:86:61    UHLW          0       13
172.16.1.3         0:80:ad:16:86:38   UHLW          0        6
172.16.255.255     ff:ff:ff:ff:ff:ff  UHLWb         0        3        ed0
192.168            10.0.0.2           UGc           2       12        de0
192.168.1          10.0.0.2           UGc           1        4        de0
192.168.2          10.0.0.2           UGc           1        0        de0
224.0.0.9          127.0.0.1          UH            1        1        lo0
```

表 7.8: 片方向通信路不通時の受信局 ulysses の経路表

```
Routing tables

Internet:
Destination        Gateway            Flags      Refs      Use      Netif Expire
10                 192.168.0.254      UGc         706      714        vx0
127                127.0.0.1          URc           0        0        lo0
127.0.0.1          127.0.0.1          UH            0        0        lo0
192.168            link#1             UC            0        0
192.168.0.2        52:54:4c:4:30:64   UHLW          1        1        vx0    416
192.168.0.254      52:54:4c:2:22:67   UHLW          3       11        vx0    425
192.168.1          link#3             UC            0        0
192.168.1.2        0:40:b4:80:34:3f   UHLW          0      653        ed1    503
192.168.2          192.168.0.2        UGc           0        0        vx0
224.0.0.9          127.0.0.1          UH            1        5        lo0
```

表 7.9: 片方向通信路不通時の aquarius から icarus への traceroute

```
aquarius % traceroute icarus
traceroute to icaurs.kirc.wide.ad.jp(192.168.1.2): 1-30 hops, 38 byte packets
 1  zodiac (10.0.0.2)  1.004 ms  0.760 ms  0.699 ms
 2  ulysses (192.168.0.1)  1.531 ms  1.284 ms  1.248 ms
 3  icarus (192.168.1.2)  2.163 ms  2.091 ms  1.998 ms
```

表 7.10: 片方向通信路不通時の icarus から aquarius への traceroute

```
icarus % traceroute aquarius
traceroute to aquarius.kirc.wide.ad.jp(10.0.0.1): 1-30 hops, 38 byte packets
 1  ulysses (192.168.1.1)  1.532 ms  1.506 ms  1.376 ms
 2  zodiac (192.168.0.254)  1.962 ms  1.938 ms  1.938 ms
 3  aquarius (10.0.0.1)  2.525 ms  2.552 ms  2.417 ms
```

に対して traceroute を行った結果を表 7.10に示す。

7.3 動的仮想リンク設定機構の性能評価

本論文で提案するの動的仮想リンク制御機構の性能を評価した。

7.3.1 評価のための動作パラメータの設定

評価にあたって、送信局 aquarius は表 7.11の通り、受信局 ulysses, valkyrie は表 7.12の通りパラメータを設定した。

表 7.11: 送信局 aquarius の設定

HELLO 送出間隔 (interval)	10 秒
仮想リンク使用有効時間 (LeaseTime)	1800 秒
IP アドレス自動設定	あり

表 7.12: 受信局 ulysses, valkyrie の設定

仮想リンク更新開始時間 (ref_border)	有効期限の 10 秒前から
メッセージ受信タイムアウト (ta_timeout)	30 秒
片方向通信路タイムアウト (udl_timeout)	30 秒
HELLO シークエンスの敷居値 (seq_threshold)	10

表 7.13: 受信局における仮想リンク確立の所要時間

測定回数	時間 (秒)
1 回目	8
2 回目	5
3 回目	1 秒以下
4 回目	1 秒以下
5 回目	2

7.3.2 仮想リンク設定動作の性能評価

受信局が受信局モジュールの実行を開始してから、仮想リンクが確立し片方向通信路が使用できるようになるまでにかかる時間を計測した。

図 7.1のネットワークにおいて、送信局 aquarius 上で送信局モジュールを実行状態にしておき、受信局 ulysses 上で受信局モジュールの実行を開始してから仮想リンクが使用可能になるまでにかかった時間を計測した。単位は秒とし、測定の精度は 1 秒とした。この結果を表 7.13に示す。

表からわかるように、仮想リンクの確立にかかる所要時間は、概ね 10 秒以内である。受信局モジュールは、実行開始してから HELLO メッセージを受信するまで待ち、これを受信したら仮想リンクの設定を開始する。送信局の HELLO 送信間隔が 10 秒であることが要因である。この結果から、HELLO の送信間隔を短くすれば仮想リンクの確立にかかる時間を短縮できることがわかる。

7.3.3 受信局における送信局の動作停止の検知

動的仮想リンク制御機構が安定動作している状態で、送信局 aquarius 上で実行されている送信局モジュールを終了させ、受信局 ulysses の受信局モジュールがこれを検知するのにかかった時間を計測した。単位は秒であり、測定の精度は 1 秒とした。この結果を表 7.14に示す。

表からわかるように、検出の所要時間は概ね 20 秒〜30 秒の間である。受信局モジュールは、最後に UDL I/F が HELLO メッセージを受信してから、一定時間以上

表 7.14: 受信局における送信局モジュール停止検出の所要時間

測定回数	時間 (秒)
1 回目	26
2 回目	25
3 回目	22
4 回目	29
5 回目	21

表 7.15: 受信局における送信局リセット検出の所要時間

測定回数	時間 (秒)
1 回目	1
2 回目	9
3 回目	8
4 回目	3
5 回目	3

HELLO を受信しない場合、送信局の障害と判断する。したがって、送信局が最後の HELLO メッセージを送信してから 30 秒を経過した時点で障害を検出する。送信局の HELLO 送信間隔が 10 秒であり、受信局の片方向通信路タイムアウトが 30 秒であることが要因である。この結果から、送信局モジュールの障害検知は、HELLO の送信間隔と受信局の片方向通信路タイムアウトの設定によって決まる。したがってこれらの値に短い時間を設定すれば、送信局モジュール検知の所要時間は実用上充分な程度の時間となる。

7.3.4 受信局における送信局のリセットの検知

動的仮想リンク制御機構が安定動作している状態で、送信局 aquarius 上で実行されている送信局モジュールを終了し、再び実行した。これによって送信局モジュールはリセットされる。受信局 ulysses の受信局モジュールがこれを検知するのにかかった時間を計測した。単位は秒、測定の精度は 1 秒とした。この結果を表 7.15に示す。

表からわかるように、検出の所要時間は概ね 10 秒以内である。受信局モジュールは、起動のたびにランダムな値を HELLO シークエンス番号の初期値として使用する。受信局は、HELLO シークエンス番号が異常な変化をしている場合、送信局がリセットされたと判断する。送信局の HELLO 送信間隔が 10 秒であることが要因である。送信局のリセット検知時間は、仮想リンク確立の所要時間に等しい。したがって、仮想リンク確立が実用に充分な程度の時間で行われる場合、送信局リセット検知の所要時

表 7.16: 送信局における経路切り替えの所要時間

測定回数	時間 (秒)
1 回目	291
2 回目	277
3 回目	305
4 回目	288
5 回目	322

間も実用に充分な程度短い時間である。

7.3.5 送信局における受信局の動作停止の検知

5.3.2 で述べたように、仮想リンク使用有効時間が長い場合、受信局の障害検出に時間がかかる。したがって、受信局の障害検知には、経路情報の更新にかかる時間を計測する。

図 7.1のネットワークにおいて、経路制御プロトコルとして RIP を動作させる。動的仮想リンク制御機構が安定動作している状態で、受信局 ulysses 上で実行されている受信局モジュールを終了し、UDL I/F を通信路から切断した。この時、aquarius での経路情報において ulysses を含む受信側ネットワークである 192.168.1/24 のネットワークへの経路が切り替わるのにかかった所要時間を計測した。単位は秒とし、計測の精度は 1 秒とした。この結果を、表 7.16に示す。

表から分かるように、経路の切り替えにかかる時間は 300 秒前後である。一般的な RIP の実装では、経路の有効時間は 180 秒であり、その経路が経路表から削除されるのは有効時間が過ぎてから 120 秒後である。送信局 aquarius が動的リンク制御機構の停止した受信局 ulysses から経路情報を受け取らなくなった場合、受信局ネットワーク 192.168.1/24 への経路情報は双方向通信路からも受信しているので、そちらに経路が切り替わる。経路情報を受け取らなくなった時点から 180 秒後に経路は無効となり、更に 120 秒後に経路は削除される。したがって、経路制御プロトコルの経路を削除する時間が、送信局における受信局の障害検出の所要時間となる。

経路制御プロトコルにおいて経路が削除される時間は、各プロトコルにおいて実用に充分な時間内に経路が収束するように設計されている。したがって本機構において受信局の動作が停止した場合の回避措置は、経路制御の動作の上で実用に充分な程度の時間内に行われる。

7.3.6 本機構のオーバーヘッド

前述の性能評価から、本機構は HELLO メッセージの送信間隔と片方向通信路タイムアウトの値を短くすれば、実用に充分な性能を持つことが分かる。しかし、HELLO

メッセージ送信間隔がを短くすると、相対的に片方向通信路上に送信される制御メッセージのためのトラフィックが増加する。ここで、このトラフィックの量について考察する。

5.2.6で述べたように、HELLO メッセージは図 5.7に挙げた構成を持つ。このメッセージは固定長である。本機構は UDP によって制御メッセージを交換するので、1 回の HELLO メッセージの送信によって発生するトラフィックの量は、次の式で計算できる。

$$1\text{回の}HELLO\text{によるトラフィック量} = IP\text{ヘッダ長} + UDP\text{ヘッダ長} + HELLO\text{メッセージ長}$$

また、HELLO メッセージの送信間隔は一定であるので、単位時間あたりに HELLO メッセージ送信によって発生するトラフィックの総量が次の式によって算出できる。

$$HELLO\text{トラフィック総量} = 1\text{回の}HELLO\text{によるトラフィック量} \times (\text{単位時間長} \div HELLO\text{送信間隔})$$

例として、HELLO メッセージの送信間隔が 10 秒であるときの、1 時間あたりの HELLO メッセージによるトラフィックを計算する。

IP ヘッダ長は 20 オクテット、UDP ヘッダ長は 8 オクテット、HELLO メッセージ長は 20 オクテットなので、1 回の HELLO メッセージ送信によって発生するトラフィックの量は、$20 + 8 + 20 = 48$ オクテットである。したがって HELLO メッセージによる 1 時間あたりのトラフィック総量は、$48 \times (3600 \div 10) = 17280$ オクテットである。

このトラフィック量をビット毎秒に換算すると、38.4bits/s となる。このトラフィック量は、既存の片方向通信路の帯域に対して無視できるほど小さい。

以上のことから本機構は、片方向通信路上の通信に対するオーバーヘッドは無視できるほど小さいことがわかる。

第 8 章

結論と今後の課題

本章では、本論文の結論と今後の課題について述べる。

8.1 結論

本論文では、片方向通信路と双方向通信路組み合わせて統合的に使用するネットワークアーキテクチャを示し、このアーキテクチャが現在のインターネットにおけるトラフィックパターンに適合する効率のよいネットワークであることを述べた。

片方向通信路は既存のインターネットにおける通信技術と親和性が低いことを述べ、片方向通信路をインターネット上の通信路として使用する際に発生する問題点について論じた。またそれらの解決手法を述べた。経路制御、データリンク層アドレス通知、通信路の状態検知の問題について述べ、この解決手法を示した。

特に、既存の手法では解決されていない大規模性の問題について論じ、これを解決するために仮想リンクとインターフェースの動的な制御が必要であることを論じた。そしてこの実現に必要な機構の設計を示し、実装を行った。本論文で提案する手法が有効であることを検証するため、片方向通信路と双方向通信路を組み合わせたテストベットを構築し、評価を行った。これにより、本論文で提案する手法が大規模性を実現ものであることを述べた。また本論文で提案する機能が実用に充分な性能を持つものであることを述べた。

本論文では、従来の片方向通信路を使用するための手法にはない仮想リンクとインターフェースの動的な制御を提案した。この機構を実現することによって、片方向通信路を含むネットワークアーキテクチャにおける大規模性の問題を解決した。本論文で提案する手法は、片方向通信路を既存の通信技術上で使用するために必要な仮想リンク及びインターフェースの制御を動的に行う。したがって、片方向通信路の使用者数に対する大規模性が向上する。したがって本手法は、片方向通信路を含むネットワークを構築するのに有効である。この手法を用いることによって、より大規模性と柔軟性のある効率のよいネットワークアーキテクチャの実現できる。

8.2 今後の課題

現在インターネット上では、複数の経路制御プロトコルが使用されている。これら経路制御プロトコルの動作全てについて本機構上での正しい動作を検証するには至らなかった。本論文で提案する手法を実用してネットワークの構築を行うため、これらの検証を行う必要がある。

また、本機構を実際のインターネットに使用する際には、さらなる大規模性が要求されると考えられる。本論文で提案した手法が耐えうる規模の考察と、片方向通信路をインターネット上で用いる際の規模とそれに適応しうる大規模性を持った設計が今後の課題として要求される。

長期的な課題として、本機構の持つ設計および機構から、片方向通信路を含むネットワークを有効に動作させうる経路制御プロトコルが持つべき要件を導出し、次世代の経路制御プロトコルの設計に生かすことが望まれる。

謝辞

本研究を進めるにあたり御指導を賜りました慶應義塾大学環境情報学部教授村井純博士並びに徳田英幸博士に深く感謝いたします。また、絶えず貴重な御助言と御指導を頂きました慶應義塾大学環境情報学部講師楠本博之博士、中村修博士に、心より御礼を申し上げます。

本研究を進める多くの段階におきまして終始貴重な御助言を頂きました (株) 日本サテライトシステムズの泉山英孝氏、(株) ソニーコンピュータサイエンス研究所の出水法俊氏、ソニー (株) の藤井昇氏、並びに WIDE プロジェクトの方々と WISH-TF の方々に深く感謝いたします。

本研究を進める上で終始惜しみない御助力を頂きました慶應義塾大学政策メディア研究科西田佳史氏、クスタルト・ウィドヨ氏、余語徹郎氏に深く感謝致します。また、本論文の執筆に当たりまして常に励ましと御協力を頂きました慶應義塾大学環境情報学部徳田・村井・楠本・中村研究室の諸兄に感謝致します。

参考文献

[1] Walid Dabbous, Emmanuel Duros, Thierry Ernst “Dynamic Routing in Networks with Unidirectional Links”, Proceedings of the Second International Workshop on Satellite-based Information Services, Budapest, Hungary, Oct. 97.

[2] Hidetaka Izumiyama, Akihiro Tosaka, Akira Kato “Uni-directional Link Routing with IP tunneling approach”, Internet-Draft, July 1997.

[3] 西田視磨, 楠本博之, 村井純, “単一方向衛星回線を含むネットワークの為のアドレス変換機構を用いたネットワークアーキテクチャ”, 電子情報通信学会和文論文誌 B-II 衛星通信実験小特集号, January 1998.

[4] Postel, J. “Internet Protocol”, RFC 791, September 1981.

[5] J. Postel, J. Reynolds “A Standard for the Transmission of IP Datagrams over IEEE 802 Networks”, RFC 1042, February 1988.

[6] Plummer, D. “An Ethernet Address Resolution Protocol - or - Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware” RFC 826, November 1982.

[7] K. Egevang, P. Francis “The IP Network Address Translator (NAT)” RFC 1631, May 1994.

[8] P. Srisuresh, K. Egevang “The IP Network Address Translator (NAT)” Internet-Draft, September 1997.

[9] “IP Spoofing Attacks and Hijacked Terminal Connections” CERT Advisory CA-95:01 January 1995.

[10] Stan Hanks, Tony Li, Dino Farinacci, Paul Traina, “Generic Routing Encapsulation (GRE)” RFC 1701, October 1994.

[11] William Allen Simpson, “IP in IP Tunneling” RFC 1853, October 1995.

[12] Charles Perkins, “IP Encapsulation within IP” RFC 2003, October 1996.

[13] Charles Perkins, "Minimal Encapsulation within IP" RFC 2004, October 1996.

[14] C. Hedrick, "Routing Information Protocol" RFC 1058, June 1988.