第 9 部

衛星通信によるネットワーク構築実験

第 1 章

研究の背景と目的

ディジタル技術の発達とともに、衛星回線を用いた片方向の高速伝送が安価に提供できるようになってきており、この技術を用いて、PerfecTV!をはじめとするディジタル衛星放送が一般家庭にも広く普及してきている。このような技術やハードウェアは、IP パケットのような衛星放送以外のディジタルデータの伝送にも利用できることから、衛星回線を用いた Internet サービスが計画、提供されてきている。

衛星回線は、基本的には片方向回線 (UDL:UniDirectional Link) ではあるが、有線を基礎にした地上回線にはない魅力的な特徴をもった伝送路である。衛星回線は、1 つの衛星回線で広地域にひろがった地域と通信できる広域性、1 つの衛星回線で、同時に複数の人と通信できる同報性、ネットワークのトポロジを動的に変えられる柔軟性を持っている。しかし Internet では、使用する回線は、基本的に双方向回線 (BDL:BiDirectional Link) であることを前提にプロトコルが設計されているため、片方向の衛星回線はそのままではうまく利用できない。このため、現状では衛星回線等の片方向回線を利用する場合には、管理者が静的な経路制御テーブルを維持管理し経路制御を行う場合が多い。その結果、片方向回線は、一定の制約のもとでの使用にとどまっている。それは、動的経路制御が必要ない小規模なネットワーク内での利用、単に選択的な受信のみの Multicast IP address(または broadcast IP address) を使用するアプリケーションの利用、である。また、UDL の障害時に自動的に代替経路へ迂回できない問題がある。

本年度は、このような問題を解決し、衛星回線のような片方向回線に動的経路制御を提供する技術 (UDLR:UniDirectional Link Routing) の研究を行った。本研究により、UDL を含むネットワークにおいて動的な経路制御プロトコルを動作させ、経路を有効に使用できるようになった。

2章では、UDLR 技術の用語、問題点および解決方法について述べる。3章および 4章では、2 つの独立した実装について述べる。

第 2 章

UDLR 技術

2.1 概要

本年度の WISH の解決手法は、UDL に直接接続しているノードが双方向のリンクに接続されているかのごとく通信が行えるようにすることにより、動的な経路制御を行えるようにするものである。本解決手法では UDLR を提供するための機構は、UDL に直接接続されたノードのデータリンク層で実現される。これにより、上位のレイヤは UDL を意識せずに、既存の経路制御プロトコルをそのまま使えるようになる。

また、UDL 上のノードの追加や削除が動的に行えるようにするための仕組み (DTCP:Dynamic Tunnel Configuration Protocol) も提供する。

2.2 用語とトポロジ

UDL にパケットを送信できるノードを Feed、UDL からパケットを受信できるノードを Receiver と呼ぶ。Feed と Receiver は、UDL で接続されている。Feed と Receiver は UDL 以外に Ethernet、PPP などのインターフェースを持っており、これを利用して互いに通信できるものとする。図 2.1は UDL 上に一つの Feed と複数の Receiver が存在する、衛星通信では一般的なトポロジを示したものである。図 2.1において、f1u は Feed の UDL Interface(Send-Only) の IP address、f1b は Feed の Internet(IP network) に接続された BDL Interface の IP address、r1u、r2u は Receiver 1、2 の UDL Interface(Receive-Only) の IP address、r1b、r2b は Receiver 1、2 の Internet に接続された、BDL Interface の IP address を表す。

2.3 UDL の問題点

Internet 上のほとんどのプロトコルでは、通信回線の双方向性が想定されている。特に、直接接続されたルータ間で使われる経路制御プロトコルは、UDL 上では正常に動作しない。実際に、Receiver は経路情報の要求や返答、経路制御の各種メッセージを、受信専用インターフェースから Feed に対して送信できない。

ネットワーク層は、データリンク層が双方向であることを前提としてアクセスする。送出されるデータグラムに対して、ネットワーク上の正しい次ホップが経路制御テーブルより選択され、適切なデータリンク層に渡される。

図 2.1では、Receiver 1 は Feed を同一のネットワークに接続されているノードとみなす。しかし、'ping f1u'を Receiver 1 で実行しても Feed から返答は得られない。Recviver 1 のネットワーク層は、受信専用インターフェースのデータリンク層にパケットを渡す。このインターフェースは、Feed に対してデータを送れない。ネットワーク層から受信専用インターフェースのデータリンク層に渡される、データグラムはすべて捨てられる。

2.4　同報型双方向ネットワークのエミュレーション

衛星回線等の UDL は、通常の通信形態として Feed から Receiver への通信を提供する。

Feed はある特定の Receiver に対して、ユニキャストのパケット、あるいはすべての Receiver に対してのマルチキャストまたはブロードキャストのパケットを送れる。しかし、それ以外の通信はできない。Receiver は他の Receiver あるいは Feed にパケットを送れない。

このような、UDL を含むネットワークにおいて、UDL の双方向の接続性を仮想的に実現するデータリンク層トンネリングメカニズムは、次の章で述べる。このデータリンク層トンネリングメカニズムの機能を定義するために、ここでは、同報型ネットワークの特性をいくつかのシナリオに分けて定義する。

同報型ネットワーク上で可能なシナリオは、次のように整理できる。

- シナリオ 1
  Receiver は Feed にユニキャストのパケットを送れる (Receiver から Feed への point-to-point の通信)
- シナリオ 2
  Receiver は、UDL 上の全てのノードに対して、マルチキャスト、ユニキャストのパケットを送れる (Receiver からの point-to-multipoint の通信)
- シナリオ 3
  Receiver は他の Receiver にユニキャストのパケットを送れる (Receiver から他の Receiver への point-to-point 通信)
- シナリオ 4
  Feed は、UDL 上の全てのノードに対して、マルチキャスト、ユニキャストのパケットを送れる (Feed からの point-to-multipoint 通信)
- シナリオ 5
  Feed は Receiver にユニキャストのパケットを送れる (Feed から Receiver への point-to-point 通信)

シナリオ 4 と 5 が、UDL 上での通常の通信である。

データリンク層トンネリングメカニズムは、これらのシナリオを実現するための手段を提供する。

このデータリンク層トネリングメカニズムは、ネットワーク層以下で動作する。これは、BDL をエミュレートするためである。これはネットワーク層に対して下位層を意識させないようになっている。データリンク層はそれが BDL であるかのごとくネットワーク層に対して振るまう。

2.5 Dynamic Tunnel Configuration Protocol (DTCP)

Receiver はカプセル化されたデータを転送するため、Feed のトンネルの出口を知らなければならない。トンネルは以下のイベントを処理するために動的に設定され、管理されなければならない。

1. 新しい feeder が立ち上がると、すべての Receiver はその Feed と通信するための双方向のトンネルを作成する。新しい Feed がいつでも UDL に接続される可能性があるので、静的なトンネルの設定は適切ではない。

2. UDL がダウンした時、Receiver はトンネルを使用不可にしなければならない。UDL が使用不可になった場合、Feed はトンネルを経由に Receiver からデータグラムを受け取ってはならない。実際にデータを受信してしまうと、通信路が利用可能であると誤って認識するいくつかの経路制御プロトコルが存在する。

3. Feed が突然使用不可能になった時、Receiver はトンネルを使用不可にしなければならない。これは、不必要なデータグラムがトンネルを使用した場合の通信のオーバーヘッドを防ぐためである。例えば Receiver は、エンドポイントが使用不可能であるトンネルに対して、ブロードキャストメッセージを転送する必要はない。

DTCP は、Receiver が Feed を発見し、使用可能なトンネルのエンドポイントの一覧を動的に管理するための機能を提供するプロトコルである。これは Feed が定期的に UDL 上に送信する、トンネルのエンドポイントのアドレスの情報を含んだ報告を基本としている。Receiver はこの報告を受信し、存在する Feed を認識しトンネルのエンドポイントの一覧を管理する。

2.6 経路制御プロトコルの設定

これまで述べてきたように、データリンク層トネリングメカニズムは、ネットワーク層と上位層から Feed と Receiver が UDL に接続されているということを隠している。通信は双方向であるが、非対称な帯域幅と遅延がある。

インターネットで UDL と BDL を統合して利用するために、Feed と Receiver は経路制御プロトコルを動作させなければならない。Feeder と Receiver は相互に直接接続されているように認識するため、正しく動作する。そして UDL を通して経路制御情報を交換することができる。

しかし、経路制御プロトコルは、通信路の片方向性を考慮しなければならない。経路制御は、ネットワーク層でインターネットの実際のトポロジと合致しなければならない。トンネリングメカニズムは、通常の通信のデータグラムを転送することが目的ではない。Receiver から Feed への経路制御情報の転送のみに利用されなければならない。Receiver 上の経路制御プロトコルは、通常の通信において受信専用インターフェースを使用しないようにしなければならない。

したがって、経路制御プロトコルの設定には、Feed から Receiver へは低いコスト、Receiver から Feed へは高いコストを設定する必要がある。同様の目的で、例えば RIP を利用する場合には、Receiver は到達可能なサブネットを Feed へ通知するが、Receiver は Feed からの通知を利用しないようにするために、Receiver の BCE 層で Receiver からの RIP パケットを破棄してもよい。

2.7　WIDE における実装

以降の章では、藤井昇、西田視磨による 2 つの独立した実装について、その詳細を述べる。

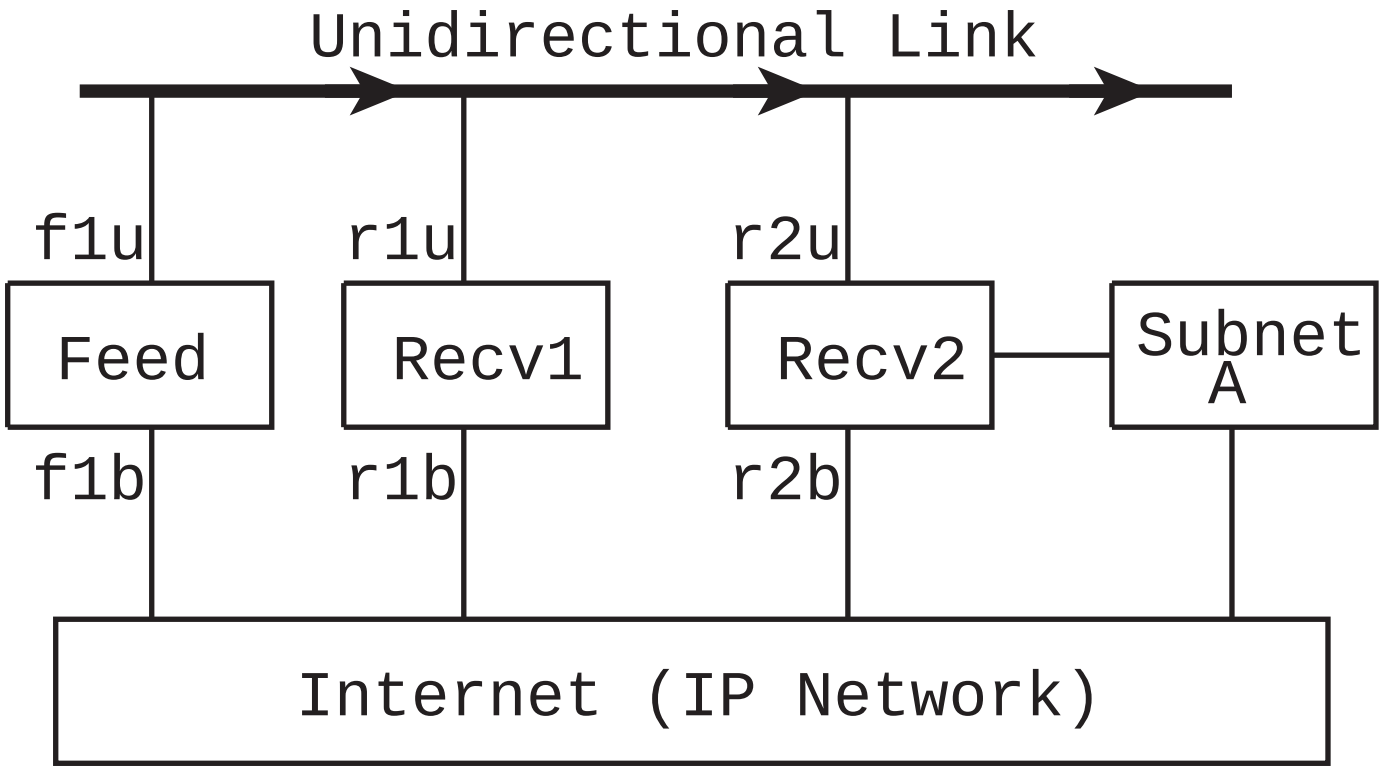

図 2.1: UDL の一般的なトポロジ

第 3 章

UDLR の実装 (1)

3.1 概要

本章では、藤井昇による UDLR の実装について述べる。以上で述べた UDLR の機能を検証するため、Feed と Receiver 上でシナリオ 1 からシナリオ 5 までの機能を実装し、動作検証実験を行った。

本実装には FreeBSD 2.2.5 を用い、Unix のカーネル内に UDLR 機能の追加した。図 3.1 に、Feed と Receiver 上の各層でのパケットの流れを示す。

3.2 Receiver での処理

Receiver 上では、ip_output() 内で経路制御表が検索され、UDL に IP データグラムが出力されることになると UDL I/F の sat_output() が呼び出される。sat_output() では、上位層から受け取った IP データグラムに GRE ヘッダと IP ヘッダを付加し、もう 1 度 ip_output() を呼び出す。この IP データグラムの宛先アドレスは Feed の BDL IP アドレスにする。するともう 1 度経路制御表が検索されると今度は BDL I/F 側が選択され、ether_output() が呼び出されるようになる。それ以降は通常の処理が行われ、BDL I/F からデータが送出される。

Receiver の実装のポイントをまとめると以下のようになる。

1. Receiver が UDL 宛または Multicast の IP データグラムを送信する場合、IP カプセリングを行い、Feed の BDL I/F 宛に送信する。

2. Receiver が BDL I/F から Unicast の Transit パケットを受信した場合、その IP データグラムを破棄する。

3. Receiver では、自分が出した IP データグラムを UDL から受信した場合、その IP データグラムを破棄する。

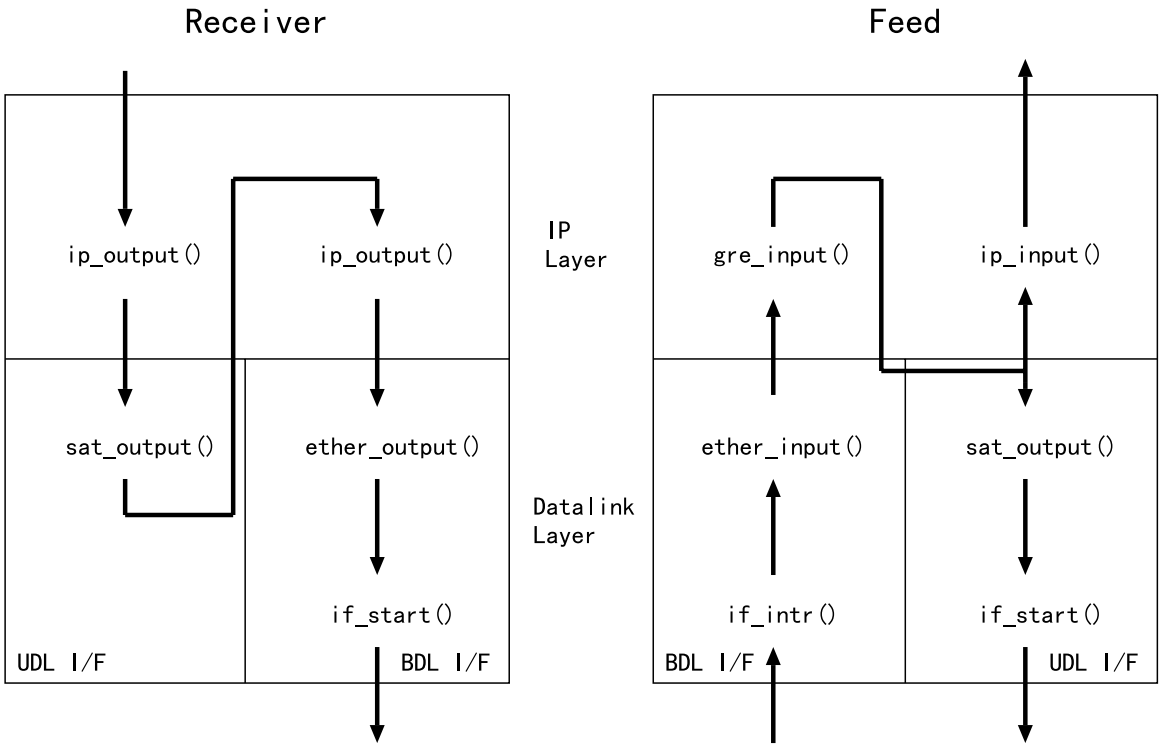

図 3.1: BCE の実装

3.3 Feed での処理

一方、Feed は BDL I/F から GRE のデータを受け取ると、まず GRE のパケットであることを認識し、gre_input() を呼び出す。gre_input() 内では、IP ヘッダと GRE ヘッダを取り除き、inner IP の宛先 IP ごとに以下にあげる処理の場合分けを行う。

1. Feed の UDL I/F のアドレスの場合
   IP 層に上げる。

2. Receiver 宛の Unicast アドレスの場合
   UDL にのみ送信する。

3. Broadcast アドレスまたは Multicast アドレスの場合
   IP 層に上げると同時に UDL にも送信する。

4. 上記以外の場合
   IP データグラムを破棄する。

以上のような処理を行うことにより、Feed と Receiver が接続された UDL が同報型のネットワークであるかのように見せかけることが可能になる。

3.4　UDLR の実験結果

以上に述べた UDLR の理論と実装が実際のネットワーク環境で動作するかを検証する実験を行った。ここでは挙動が分かりやすい RIP の場合について述べる。

UDLR の実装を載せた Feed と Receiver と通常なルータを使用し、図 3.2に示す実験ネットワークを構成した。ここで、Receiver は Receiver 間通信も検証する必要があるため、2 台の Receiver を用意した。

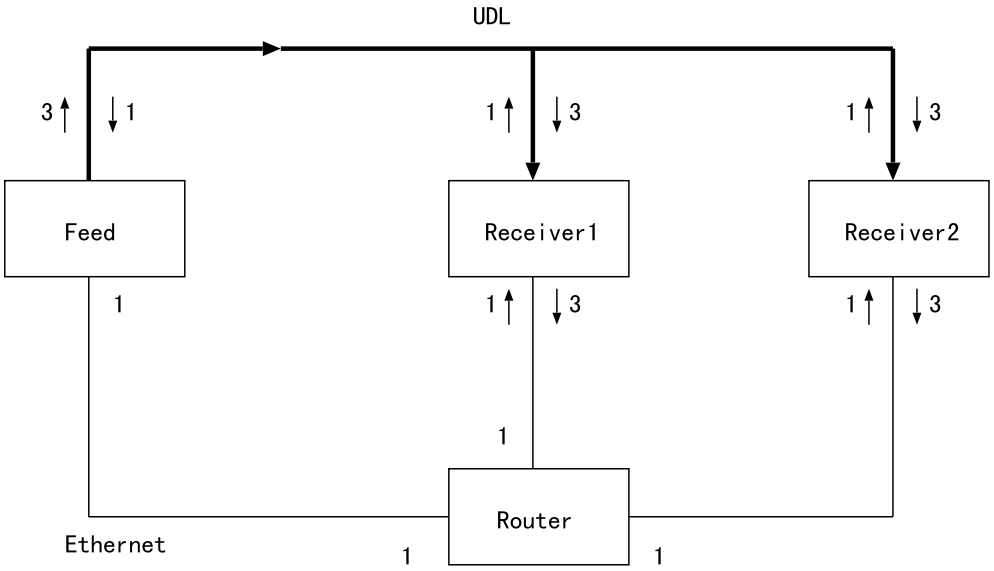

図 3.2: RIP の検証実験

今回、経路制御デーモンとして gated を使用したが、gated は、経路情報を送受する際の metric の設定を in 方向と out 方向で別々に設定できる。図 3.2の矢印は、各インターフェースに設定した in 方向と out 方向を示し、付加されている数字は metric の値を示している。

Feed は、Receiver が Feed 付近のセグメントに対する通信を行う際に UDL を使わないように、UDL に対して大きな metric をアナウンスする。Receiver は、Feed が Receiver 付近のセグメントに対する通信に UDL を積極的に使うように低い metric をアナウンスする。しかし、このような設定だけでは、Receiver 2 が Recever 1 に対しパケットを送ろうとすると metric の小さい UDL を使おうとしてしまうという不具合が起こる。そこで、この場合には Receiver の in 方向の metric に大きな値を設定することで、UDL IF から受信した RIP の経路の metric は大きいものと認識される。この結果、Receiver から UDL 以外のネットワークに対する通信には BDL が使われるようになる。

この他にも RIP2、OSPF、DVMRP の環境を構築し、動作を確認した。今後は、BGP4 や PIM-SM などのルーティングプロトコルを使用した場合の検証実験を行う予定である。

第 4 章

UDLR の実装 (2)

4.1 概要

本章では、西田視磨による UDLR の実装について述べる。片方向の通信路を含むネットワークに於いて動的な経路制御プロトコルを動作させるため、動的な仮想リンク制御機構を実装した。

4.2 設計

本機構は、仮想リンク部と制御部に分かれる。
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。

制御部は、仮想リンク部をリンクの状態に応じて制御する機構を提供する。送信局と受信局の間で相互にメッセージを交換し、各々の状態に応じてトンネルの設定や解除を行う。

4.2.1 仮想リンク部の設計

仮想リンク部は、受信局側カプセル化モジュールと、送信局側脱カプセル化モジュールによって構成される。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 4.1に示す。受信局側において、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。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。送信局側においてデータリンク層 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 フィールドを検査したのち、パケットが受信局でカプセル化されたものであった場合、これを脱カプセル化する。

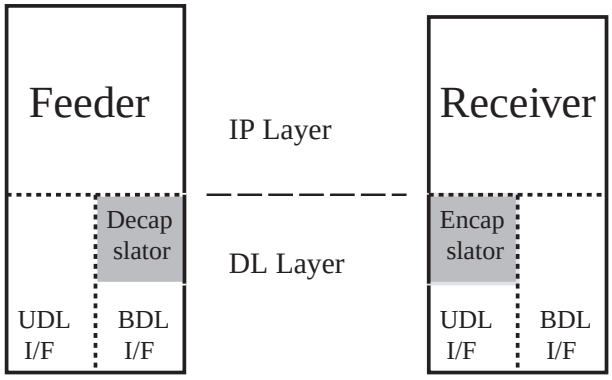

図 4.1: モジュール実装層

4.2.2　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。

4.2.3　本機構の基本的動作

本機構は、送信局と受信局が 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。本機構の基本的な動作を次に述べる。

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。

これらメッセージの交換手順を図 4.2に示す。

4.2.4　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、送信局の障害、受信局の障害、通信路の障害の 3 つが挙げられる。

- 送信局の障害

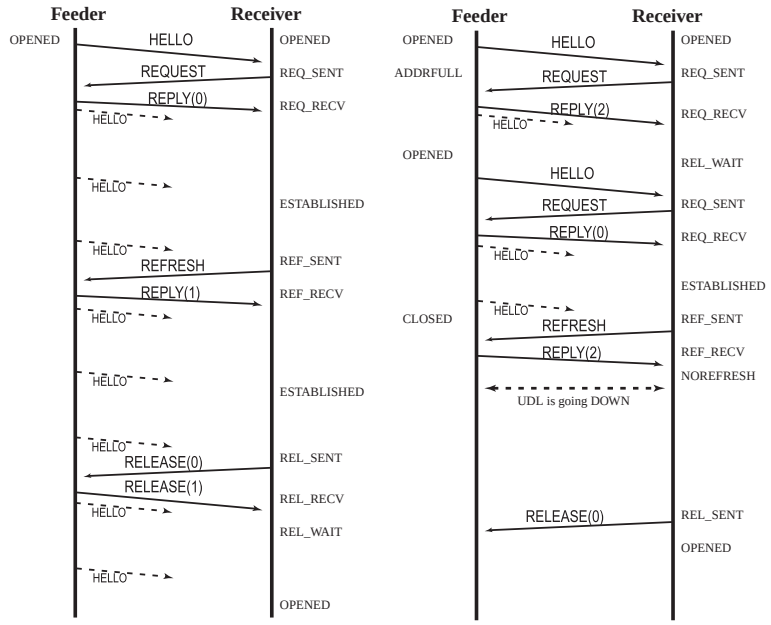

図 4.2: メッセージ交換図

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検知できる。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 を切り離せる。

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検知できる。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検知できる。

- 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

検知できる。

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。

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検知できる。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。

- 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。

4.3　実装

実装には、FreeBSD 2.2.1-RELEASE を用いた。双方向通信路には Ethernet を使用し、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。

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。ifnet 構造体と ifreq 構造体がトンネリング機構のための情報を保持、操作するように改造した。トンネル情報の読み出し、書き込みには ioctl ルーチンを使用した。

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。

4.4　評価

本機構の評価のため、図 4.3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。経路制御デーモンには、gated Revision3.5 Beta3 を用いた。

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 [103] を使用した。

評価項目は、次に挙げる 2 つである。

1. 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。

2. 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。

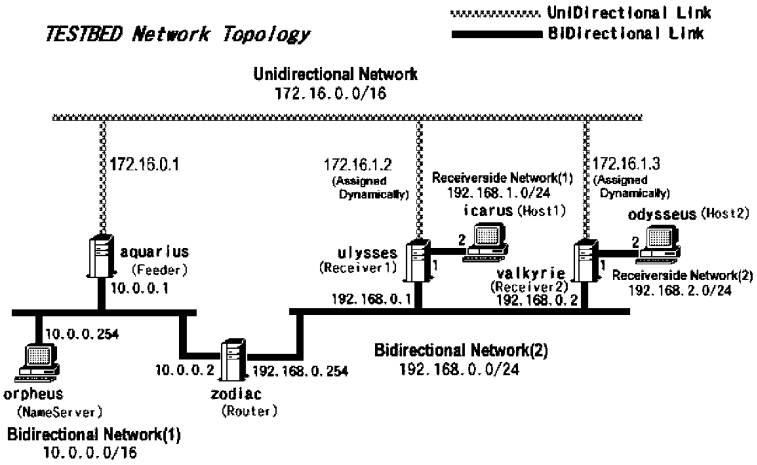

図 4.3: テストベッドネットワーク・トポロジー図

4.4.1 経路の設定

図 4.3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。

4.4.2　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 4.1, 表 4.2に示す。

表 4.1: 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 4.2: 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。

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 4.3 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 4.4に示す。

表 4.3: 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 4.4: 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
```

4.4.3 片方向通信路が不通である場合の経路

前述 4.4.2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 4.5、表 4.6 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

表 4.5: 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 4.6: 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
```

上で icarus に対して traceroute を行った結果を表 4.7に、icarus 上で aquarius に対して traceroute を行った結果を表 4.8に示す。

表 4.7: 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 4.8: 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
```

以上から、UDL を含むネットワークにおいて本機構を使用し動的な経路制御を行った場合、UDL が使用可能である場合はこれが使用され、使用不可能である場合は代替経路へと経路が切り替わることが確認された。

第 5 章

まとめ

WISH WGでは、UDLR技術に関してIETFでの議論に積極的に参加し、複数の実装を基にインターネットドラフト作成に貢献した。WISH WGは初期のころからUDLを仮想的にBDLに見せるようにし、既存の動的経路制御プロトコルを変更せずに、広域の同報型通信媒体としてUDLをうまくインターネットアーキテクチャに取り込むことを主張し、実験してきた。今年度は、衛星通信を用いた実装と実験を行い成果をあげた。本技術は、衛星通信に限らず、一般的なUDLに適用可能であり、汎用的な技術として、価値がある。

また、DTCPのような、仮想ネットワーク形成のための動的なトンネル設定プロトコルに関しても研究が進み、実験と評価を行う段階に達した。

今後は、大規模な環境での実験と動的なトンネル設定の枠組に関して、研究を進めていく。また、広域同報型の通信媒体に適したアプリケーションの構築も課題である。