第 4 部

マルチキャスト通信

第 1 章

はじめに

マルチキャスト通信ワーキンググループでは、近年重要になってきている IP マルチキャストを用いた通信技術について研究・実験を行なっている。

広域分散システムにおいては、電子掲示板システムや一般に公開されたファイルの転送といったことにより多くの情報が共有されている。また、実時間的な会話型応用ソフトウェア、メーリングリストによる電子メールのグループへの配送といった情報共有や情報配送が非常に活発に行なわれており、インターネットのような広域分散システムにおいて非常に重要な機能となっている。

これらの機能を効率良く実現するために、広域性のある経路制御機構、また、実時間性を必要とする音声・画像通信のための、信頼性のある遅延のばらつきの少ない通信のための資源予約機構、広域のマルチキャスト型通信媒体としての衛星通信機構について研究を行なっている。さらに、マルチキャスト通信を有効に利用した 1 対多型の通信に適したアプリケーションとして、ニュースやメーリングリストの配送、FTP アーカイブの転送といったものや、マルチキャスト archie のような分散検索システムの研究も行なっている。

本稿では、1994 年度、マルチキャスト通信ワーキンググループで行なった次のような研究・実験について報告する。

- マルチキャスト型情報検索ソフトウェアの構築
- 資源予約型マルチキャストトランスポートプロトコルの研究
- MBONE の運用と管理、運用ガイドの作成

第 2 章

マルチキャストを利用した情報検索システムに関する研究

2.1 はじめに

インターネットの普及に伴い、多くの情報がインターネット上で提供されるようになった。さらに、インターネット上で情報提供するためのアプリケーションの発達によって、誰でも簡単にインターネット上で情報を提供することが可能になってきている。これを情報を利用する側から見ると、大量の情報がインターネット上のいたる所に散乱していて、自分の求めている情報を見つけるのが困難になっている。これらの情報を検索するには、求める情報の種類に応じた情報検索システムを利用する必要がある。

既存の情報検索システムは、情報の内容や量があまり変化しない静的な情報を検索するものである。一方、World-Wide Web[22] のように誰でも簡単に情報を提供できるシステムでは、大量の情報が広い範囲に分散するとともにその量も爆発的に増加し、動的に構造が変化する傾向がある。そのため、既存の情報検索システムで検索することは難しい。

本論文では、このように散乱した情報を「超分散情報」と定義した。超分散情報は従来の検索システムでは検索することができない。そこで、一対多通信であるマルチキャストを利用した超分散情報検索システムを提案する。さらに超分散情報の一つ anonymous FTP[23] 上で提供されているファイルを検索するシステム「march」の実装について報告するとともにその評価を行なう。

2.2節では既存の情報検索システムについて説明し、それらの特徴を述べる。2.3節では march の実装について説明する。2.4節では march で使う検索や通信の方法について評価を行なう。

2.2 情報提供・検索システム

2.2.1 情報提供・検索システムの特徴

情報提供・検索システムは以下のように分類できる。

- システムの役割
  - 情報の提供
  - 情報の検索
  - 情報の検索および提供
- システムの形態
  - 集中型
  - 分散型

情報の提供のみを行なうシステムとしては、gopher、World-Wide Web が挙げられる。これらは指定された情報を利用者に提供する。

情報の検索のみを行なうシステムとしては、ドメインネームサーバ、archie が挙げられる。ドメインネームサーバは目的である計算機のインターネットアドレスを検索し、archie は目的であるファイルを提供している anonymous FTP ホストとディレクトリを提供する。

情報の検索と提供を行なうシステムとしては、whois、WAIS、OSI ディレクトリサービスが挙げられる。whois は NIC で提供される情報、WAIS は文書、OSI ディレクトリサービスはさまざまな内容を含むエントリを検索して提供する。

集中型システムは一つのサーバで提供する、もしくは検索する全てのデータを保持して、他のサーバとは協調しない。分散型システムは複数のサーバで分担して情報を保持し、利用者からの提供・検索要求に対して複数のサーバの検索結果を返す。分散型システムはさらに、木構造によってデータの分散を行なうシステムと、網構造によってデータを分散するシステムに分けることが可能である (図 2.1、図 2.2、図 2.3)[24]。それぞれの特徴を以下に述べる。

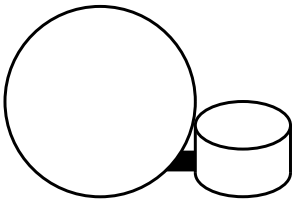

図 2.1: 集中型

集中型情報検索システム

集中型システムは whois, archie, WAIS によって代表されるシステムであり、一つの計算機で全てのデータを保持する。特徴を次に挙げる。

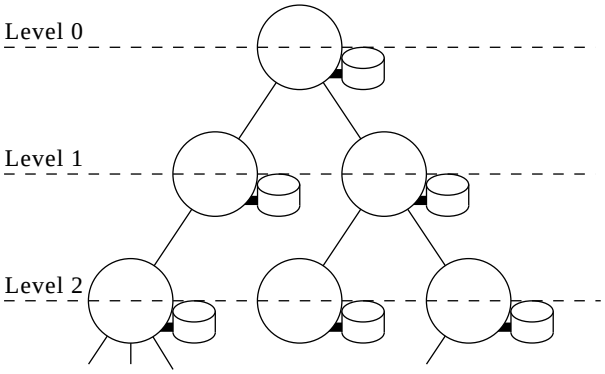

図 2.2: 木構造による分散型

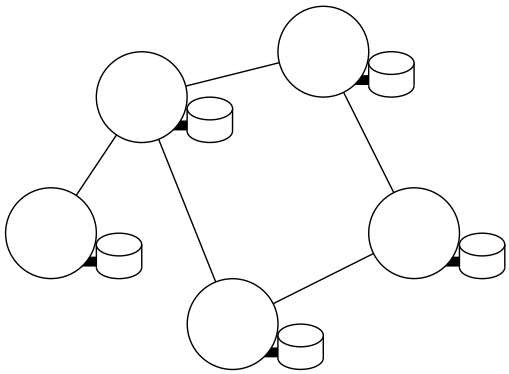

図 2.3: 網構造による分散型

長所
- データの一貫性を保つことが容易。
- 登録や更新が容易。
- 全数検索が容易。

短所
- 検索によって計算機の負荷が大きくなり易い。
- アクセスの集中によって回線が混雑し易い。
- システム障害、もしくは利用者からシステムまでの回線のどこかの障害によって、利用者はシステムを利用できなくなる。
- 検索対象の情報はその管理人によって保守されており、誰もが保守できるわけではない。
- データの集中によりサーバに大きな負荷がかかり易い。

これらの欠点を補うために、インターネット上にサーバの複製を立ち上げることによってアクセスの集中を防ぐことが多い。しかしそれぞれが等しく全てのデータを持つため、データの集中を避けることは不可能である。今のところうまく機能している集中型のシステムもインターネットの拡大によって破綻する可能性がある。

分散型情報検索システム

木構造による分散型システムはドメインネームサーバ、 OSI ディレクトリサービスによって代表されるシステムであり、サーバ間やデータ間に従属関係を持つ。網構造による分散型システムは gopher, World-Wide Web によって代表されるシステムであり、サーバやデータは従属関係を持たず互いに独立である。それぞれの特徴を以下に挙げる。

長所 **［共通］**データが分散しているので、アクセスも分散され集中型よりも検索による負荷が低い。

［共通］データの保守を分散させることができるため、保守の負担を軽くできる。

［共通］データを分散させるため、利用者とサーバ間の通信による回線の負担の減少が期待できる。

［共通］分散して存在するサーバのいずれかに障害が生じてもシステム全体の機能が停止することはない。

［木構造］従属関係に合わせた検索を行なうことによって不要な検索を抑えることが可能である。

［網構造］従属関係が薄いので個人で自由に情報を提供できる。

短所 **［共通］**データが分散されるので全数検索が困難になる。

［共通］分散されたいくつかのデータに対して検索を行なう場合、ネットワーク遅延が検索速度に影響する。

［共通］データの複製が分散される場合、データの一貫性を得ることが難しい。

［木構造］サーバの新規立ち上げや停止の際に、上位や下位のサーバの管理者との連携が必要。

［網構造］検索範囲の指定が難しい。

2.2.2　超分散した情報の検索への利用

最近のインターネットでは World-Wide Web の普及にともない、画像や音声などサイズの大きなデータを使った情報を個人が自由に提供することが流行している。これらの情報は個人が自由に提供できる反面、情報間の従属関係がないため、木構造を使った分散型システムでは、利用者が必要な情報を検索することができない。また、これらの情報は爆発的に増加する傾向を持つ。そのため、集中型システムでは情報の増加速度や変化に追従していくことが難しい。

この種の情報は今のところ網構造を使った分散情報提供システムで提供されている。網構造では効率の良い検索範囲を指定することが難しいため、この種の情報から遅延や洩れのない検索を行なうシステムは存在しない。

本章では、爆発的に増加するこの種の情報を「超分散情報」と定義する。超分散情報は個人で提供することが多く、数が多く増加速度も速いことから、次の特徴を持つ (図 2.4)。

- 全ての情報の存在を認識することは不可能。
- 洩れのない検索を行なうことは不可能。
- いくつかが集まって弱いグループが形成される。弱いグループとは、「内輪にはなんとなく知られている」という場合の「内輪」に相当する。

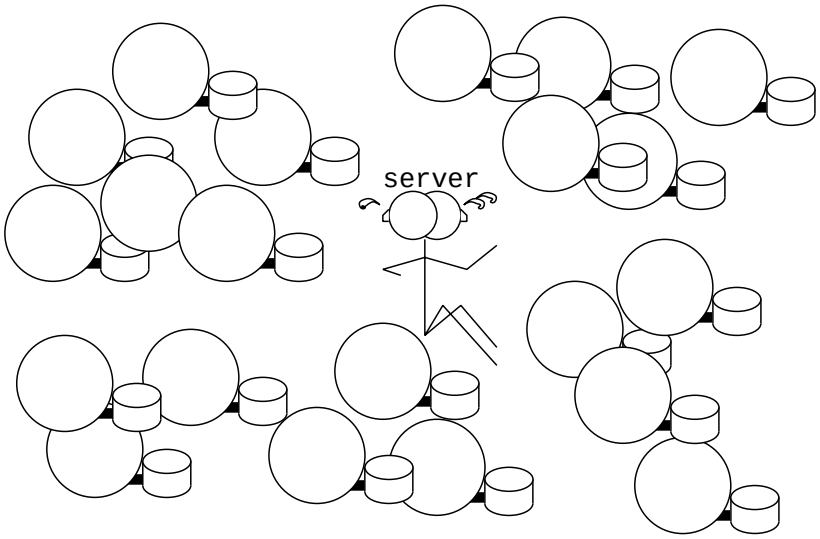

図 2.4: 超分散情報

- グループ間に明確な境界はない。

超分散情報は上に述べたように既存の情報検索システムで検索することが難しく、今は検索できてもいずれ破綻することが予想される。この問題を検索を行なうクライアントがマルチキャストを用いて、同時に複数のサーバと通信することにより解決を図ったのが本研究である。

2.3 anonymous FTP ファイル検索システム「march」

従来の情報検索システムは一対一通信を基に構築されているため、広域の検索にはさまざまな構造を導入する必要がある。そのため超分散した情報に対応できない。しかし一対多通信であるマルチキャストを利用することにより、超分散した情報に対応した検索システムが容易に構築できる。

超分散された情報は数多くの人が簡単に提供しているものが多く、複数の情報が密接な関係を持つことは少ない。しかし超分散された情報のうちのいくつかが集まって弱いグループが形成される。

この弱いグループはマルチキャストを利用した検索システムにおいて検索範囲を設定するのに利用できる。マルチキャストグループを弱いグループにマッピングすれば、利用者はマルチキャストグループに検索要求を出すことによって弱いグループ内を検索できる。

我々は、超分散情報として anonymous FTP で提供されているファイルを取り上げ、 IP

マルチキャストを利用してファイル名とその場所を検索するシステム「march」[1] を作成した。この章では march の設計方針と実装について述べる。

2.3.1　anonymous FTP と archie

anonymous FTP はサーバを立ち上げるのに特別な他のサーバや管理者との連絡や協議を必要としないため、計算機の管理者であれば誰でも立ち上げることが可能である。また近年の PC-UNIX の普及により、個人の計算機で anonymous FTP を運用することが可能になったことも要因となり、anonymous FTP はその手軽さから世界中に数えきれないほど立ち上がっている。

現在 anonymous FTP によって提供されている、ファイルを検索するシステムとして archie がある。 archie は前述したように集中型情報検索システムである。前もって複数の anonymous FTP から `ls-lR` ファイルを取得して提供されているファイルの名前を調べておき、利用者からの検索に対してマッチするファイル名を返す。

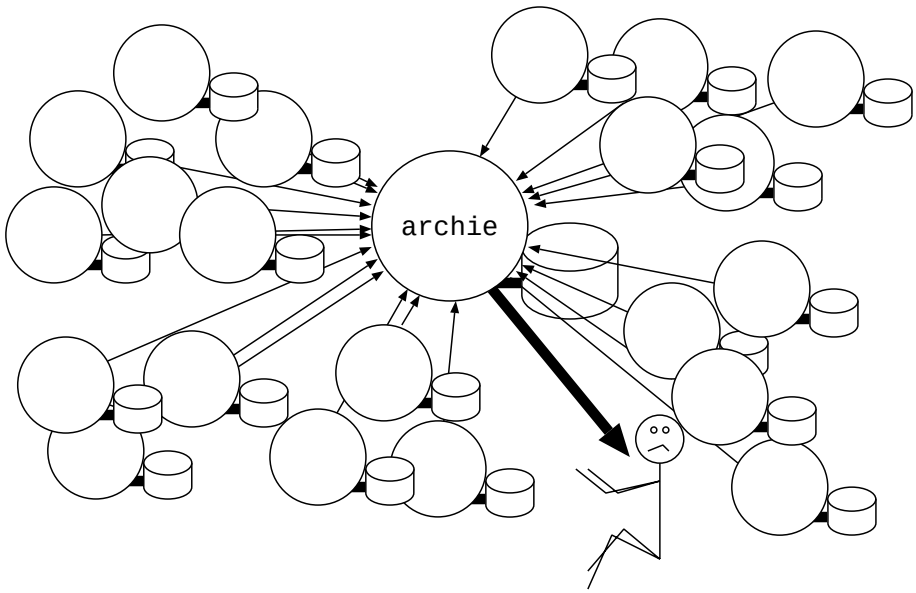

図 2.5: archie によるファイル検索

archie は集中型のシステムであるうえ、anonymous FTP で提供されるファイル数が激しく増加しているため、検索による archie サーバの負荷は非常に大きくなっている。さらに、anonymous FTP の運用開始/停止に伴う archie のサーバの設定は全て archie サーバ管理者の負担となるため、全ての anonymous FTP 上のファイルを検索することは不可能である。

[1] ftp://ftp.csce.kyushu-u.ac.jp/pub/Net/march/march-2.0.tar.gz

2.3.2 march の設計方針

march はクライアント・サーバ方式によって anonymous FTP で提供されているファイルを検索する。各サーバはそれぞれ一つの anonymous FTP で提供されているファイルの情報を保持して anonymous FTP と一緒に超分散され、IGMP によって march マルチキャストグループに参加し、クライアントからの検索要求を待つ。

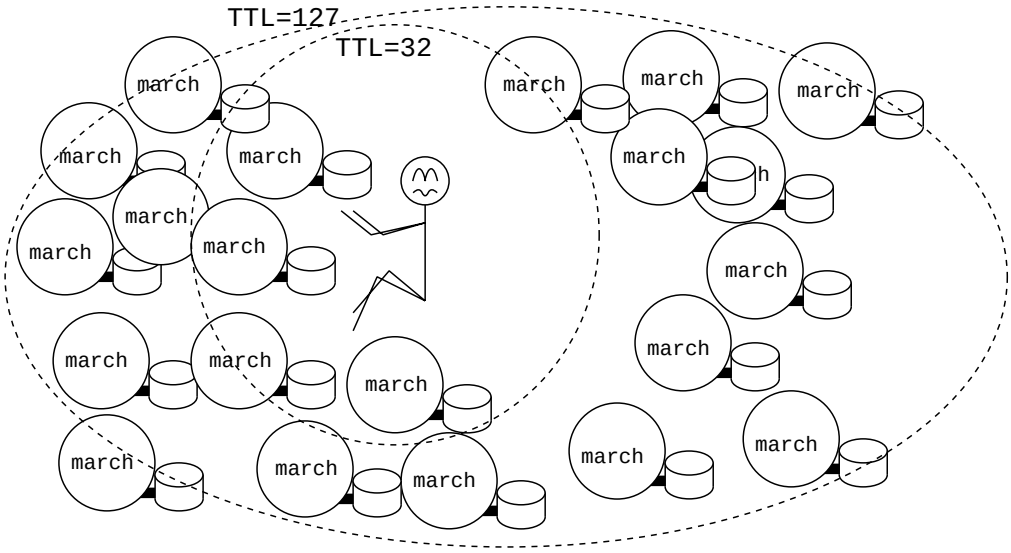

図 2.6: march によるファイル検索

検索要求は TTL と threshold によって定まる範囲内の全てのサーバに届く。サーバのないところには検索要求が届かない。

2.3.3 march の実装

プロトコル

クライアントからサーバへ送信する検索要求はサーバに渡す属性のリストから成り立ち、IP マルチキャスト UDP/IP の一つのパケットとして各サーバに送信される。各属性は属性名と属性値によって構成され、一つの検索要求で同じ属性名を持つ属性を複数持つことはできない。

クライアントからの検索要求に対するサーバの検索結果は、クライアントに対してユニキャストで返送される。クライアントは検索結果の返送に使うプロトコルとして TCP もしくは UDP のどちらかを選び、検索要求内にあらかじめ指定しておく。サーバは検索要求内にあるプロトコルに従って、検索結果をクライアントに返送する。

以下にクライアントからサーバに送信する検索要求中に使われる属性の属性名とその意味を挙げる。

VERSION march クライアントのヴァージョン。現在は 2.0。

METHOD サーバがクライアントに検索結果を返す時の通信方法。現在は TCP か UDP。TCP を使う場合はクライアントが受信に使うソケットのポート番号を PORT によって指定する必要がある。

PORT クライアントが受信に使うポート番号。METHOD=UDP の時は検索要求の送信ポートで検索結果を受信できるので必要ない。

KEY 検索に使うキー。空文字列の場合はサーバ一覧の要求。

TYPE 検索の種類。正規表現、部分文字列、シェルのファイル名展開規則の三種類による検索が可能。

DISPLAY 検索結果の表示方法。URL[25](Uniform Resource Locator) のみの要求と、ファイルに対してサーバの知っている全ての情報の要求ができる。

サーバからクライアントに返送する検索結果は、検索キーにマッチしたファイルの URL もしくはサーバの知る全ての情報で、TCP もしくは UDP によってクライアントに渡される。

また、IP マルチキャストは送信者が受信者を把握することができないので、クライアントは自分の検索要求に従って検索処理を行なっているサーバを把握できず、検索の終了する時点を判別できない。そのため、サーバは検索要求を受信したらすぐにクライアントに対して ACK を送り返す。クライアントは ACK を送り返してきた全てのサーバの検索結果が揃った時点で検索の終了と判断する。 ACK にはサーバに関する簡単な説明を含むことが可能である。

順を追って説明すると、

1. クライアントが march マルチキャストグループに対して検索要求のパケットを送信する。
2. 検索要求を受けとったサーバはクライアントに対して ACK を返送する。
3. 検索結果を受けとったサーバは、検索要求で指定された返送方法に従って検索結果をクライアントに返送する。
4. クライアントは ACK を返した全てのサーバからの検索結果を受信してユーザに返す。

検索要求パケット中の検索のキーが空文字列である場合はサーバの一覧を要求するので検索結果は返送されず、クライアントは ACK を返してきたサーバを一覧にして利用者に返す。

クライアントの動作

クライアントは利用者の要求に基づいて検索要求パケットを march マルチキャストグループに送信する。実装には C 言語を用いた。検索要求中のいくつかの属性はクライアントのコマンドライン引数で指定する。

anonymous FTP で提供されているファイルは、複数の anonymous FTP が同じファイルを複製して持つことが多いので、目的のファイルが遠くにあることは少ない。この特徴を利用して march では、はじめは小さな TTL で検索要求を送信する。目的のファイルが見つからなかったら順を追って TTL を大きくして検索要求を送信する。

サーバの動作

サーバはクライアントからの検索要求に基づいて検索を行ない、検索結果をクライアントに返す。サーバは大きく次の部分にわかれる。

受信部: クライアントからの検索要求パケットを受信し、検索部を起動する。

検索部: 受信部から起動され、内部のデータベースを利用して検索要求にマッチするファイル名の URL もしくはファイルの情報をクライアントに返す。

データベース構築部: 定期的に起動され、anonymous FTP のディレクトリ情報 `ls-lR` に基づいて検索用のデータベースを構築する。

IP マルチキャストパケットを扱う受信部の実装には C 言語を用いた。検索部と、データベース構築部のない取得した `ls-lR` からデータベースを構築する部分の実装には、文字列処理に優れた perl を用いた。データベース構築部のうち anonymous FTP から `ls-lR` を取得する部分の実装は、サーバや anonymous FTP ホストの環境に大きく依存するため、書き換えやすいシェルスクリプトを用いた。

サーバは anonymous FTP を運用している計算機上で動作し、その計算機の anonymous FTP で提供されているファイルの情報を保持して検索する。しかし、anonymous FTP を運用している計算機が IP マルチキャストに対応していない場合には、IP マルチキャストに対応している別の計算機がその anonymous FTP 上のファイルの情報を取得・保持して検索することが可能である。逆に、anonymous FTP 上で提供されているファイルの数が多過ぎる場合には、ファイルの情報を複数の march サーバで分担して保持・検索することが可能である。

クライアントは最初小さな TTL で検索要求を送信し、見つからなかった場合に TTL を徐々に大きくして同じ検索要求を送信する。このことから、あるサーバに同じクライアントから同じ検索要求が再び来た場合は、そのサーバには目的のファイルが存在していなかったことが分かる。

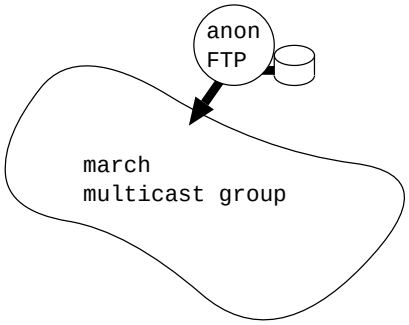

図 2.7: 通常のサーバの形態

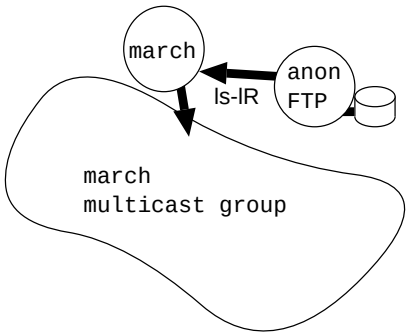

図 2.8: 代理サーバ

サーバはクライアントの IP アドレスとポート番号を記憶し、同じクライアントからの検索要求は既に答えたものとして破棄する。現在の実装では、IP アドレスとポート番号は 10 分間保持される。

代理クライアント

IP マルチキャストに対応した計算機は普及しつつある。しかし、現在の時点では全ての計算機が IP マルチキャストに対応しているわけではない。そこで march では、IP マルチキャストに対応していない計算機上のクライアントからの要求を march マルチキャストグループに転送する代理クライアントが必要となる。

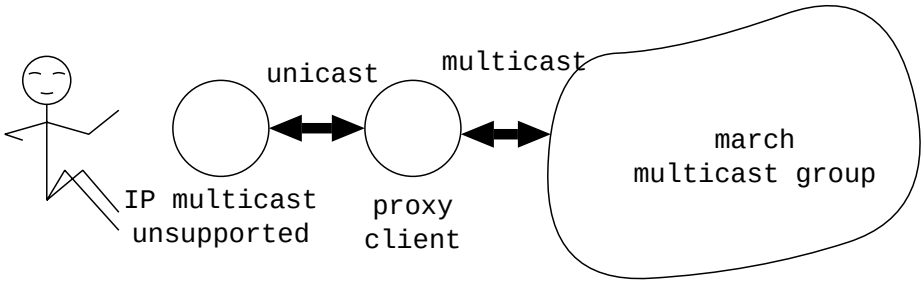

図 2.9: 代理クライアント

2.4　march の評価

現在 march のサーバは図 2.10に示すところで動作している。この情報は、クライアントを引数なしで起動することにより随時確認することが可能である。これらのサーバを利用

して、march の評価を行なった。

図 2.10: march 運用組織一覧

2.4.1 MBone との相性

現在 MBone では threshold によって、回線の太さに応じてパケットの流量を制限している。この手法を march から利用する立場から見ると、回線が太く threshold の小さなところにいる利用者からは TTL によって決まる検索範囲が回線の容量をうまく反映しているので、近くにあるファイルを効率良く発見することが可能である。

しかし回線が細く threshold の大きなところにいる利用者にしてみると、その細い回線の外を検索するためには、大きな TTL を検索要求に設定してやらなければならない。その場合、非常に広い範囲を検索範囲としてしまうことになり、さまざまな広さの検索範囲を設定することができない。

太い回線を通してつながっている場所の例として東京大学、細い回線を通してつながっている場所の例として九州工業大学を例にとる。

東京大学から検索を行なう場合は以下の中から検索範囲を選択することができるため、柔軟な検索範囲の指定が可能となる。なおここで挙げる TTL と検索範囲の対応は、1995 年 2 月 16 日現在の MBone の設定による。

$TTL < 32$ 学内。

$TTL < 64$ 768kbps 以上の回線で繋がっている所。慶応大、大阪大、京都大など。

$TTL < 96$ 512kbps 以上の回線で繋がっている所。神戸大、南山大など。

$TTL < 128$ 384kbps 以上の回線で繋がっている所。九大、佐賀大など。

$TTL < 160$ 192kbps 以上の回線で繋がっている所。九州芸工大、千葉大など。

$TTL < 192$ 96kbps 以上の回線で繋がっている所。MBone に参加している日本国内の全ての組織はこれに含まれる。また、NASA 経由で海外も検索範囲に含まれる。

$TTL < 224$ 32kbps 以上の回線で繋がっている所。

$TTL < 256$ MBone 上の全組織。

それに対して、九州工業大学から検索を行なう場合選択できる検索範囲は以下に挙げるだけしかない。

$TTL < 160$ 学内。

$TTL < 192$ 96kbps 以上の回線でつながれている所。

$TTL < 224$ 32kbps 以上の回線でつながれている所。

$TTL < 256$ 全組織。

柔軟な検索範囲を指定できない場合、ユーザは見つかったファイルのうちどれを持って来れば細い回線に負荷がかからないか分からないという問題が生じる。この問題は、本来組織内外や 地域 の範囲を設けるために使われる threshold を、回線の混雑を抑えるために流用していることが原因と考えられる。

2.4.2　ネットワークにかかる負荷

archie では一つのサーバに一度検索要求を投げることによって、そのサーバの知る中で検索要求にマッチするファイルの情報が全てクライアントに返される。一方 march では検索範囲を変えて繰り返し検索要求を投げ、検索範囲内のどれかのサーバが利用者の目的であるファイルの情報をクライアントに返した時点で、ユーザが検索を終了させる。

正規表現 “kterm” によって march で九州大学から検索を行なった結果を表 2.1 に示す。正規表現 “kterm” は X-Window 上で動くターミナルエミュレータ kterm に関係するファイルにマッチする表現である。

march による検索で全部で 51 個のファイル、重複を除くと 27 個のファイルが発見された。$TTL = 127, 191$ の検索範囲の時に多数のファイルが発見されていることから、この二つの検索範囲内の anonymous FTP のうち少なくとも一つが、正規表現 “kterm” にマッチするファイルを多数持っていることが分かる。

表 2.1: march によってネットワークにかかった負荷と見つかったファイルの数

TTL	受信 パケット数	受信 バイト数	見つかった ファイル数	新規に 見つかった ファイル	見つかった ファイル数合計
3	1 個	259 バイト	4 個	4 個	計 4 個
10	1 個	144 バイト	2 個	なし	計 4 個
31	1 個	0 バイト	なし	なし	計 4 個
95	1 個	196 バイト	3 個	2 個	計 6 個
127	3 個	1655 バイト	22 個	15 個	計 21 個
159	1 個	0 バイト	なし	なし	計 21 個
191	2 個	1702 バイト	20 個	6 個	計 27 個

しかし見つかったファイル数合計を見ると、$TTL = 127$ の時に見つかったファイル 22 個中の 15 個はそれまで見つかっていなかったファイルであるのに対して、$TTL = 191$ の時に見つかった 20 個中それまで見つかっていなかったファイルは 6 個と少ない。このことから、anonymous FTP 上のファイルを検索するためにむやみに広い範囲を検索する必要がない事が分かる。

このことから、一度に国内中、世界中を検索する archie と比べて、動的に検索範囲を指定できる march は負荷の分散、軽減の上で優れている。現在は march サーバの数が少なく TTL の値も小さくはないが、march サーバの数が増加すれば組織内、地域内程度の検索範囲で大抵のファイルは発見されるものと考えられる。

2.4.3 検索時間とファイル情報 ls-lR の量の関係

`ls-lR` のサイズとサーバの検索部がキー “emacs” で検索を行なったシステム時間のグラフを図 2.11に示す。なお時間の測定は HelioSstation1030/SunOS4.1.3 メモリ 80Mbyte によって行なった。

`ls-lR` のサイズと検索に要した時間は、それぞれ別の傾きを持った二つの直線によって近似できる。`ls-lR` のサイズを x(KBytes)、検索時間を t 秒 とすると、図 2.11中の上部領域を除いて最小二乗法によって求めた近似式は、$t = 7.35 \times 10^{-5} \times x + 6.15 \times 10^{-2}$ で表され、下部領域を除いて最小二乗法によって求めた近似式は、$t = 1.98 \times 10^{-3} \times x + 1.52 \times 10^{-3}$ で表される。複数の傾きがある原因はまだ究明中である。

また `ls-lR` のサイズの大きな所では `ls-lR` を分割して複数の march サーバに分担させることにより、検索時間を短縮させることが可能である。

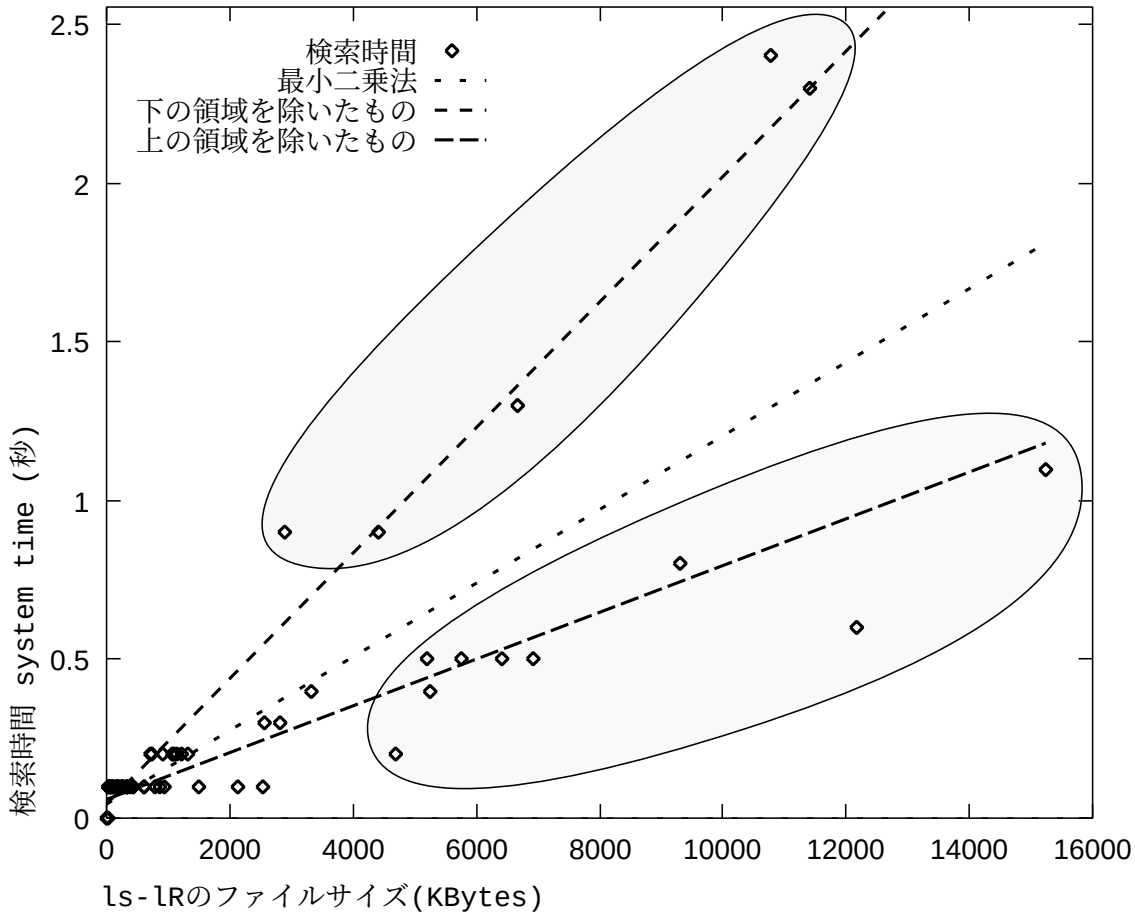

図 2.11: ls-lR のサイズと検索時間

2.4.4 検索結果の送信時間

サーバからの検索結果はユニキャストによってクライアントに返される。この時の通信方法として TCP を使った場合と UDP を使った場合にかかった時間を比較した。

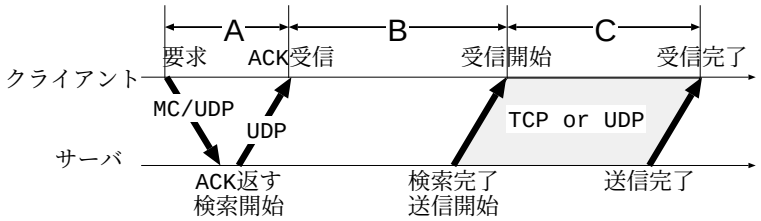

図 2.12: 各処理にかかる所要時間

図 2.12 中の各点の平均時間と最長時間を表 2.2に示す。この時立ち上がっていた march サーバの数は 7 個 であった。対比として京都大学で公開運用している archie による検索時間を表 2.3 に示す。 archie の場合は図 2.12中の B と C の区別はない。また、この時 archie サーバが管理していた anonymous FTP の数は 80 であった。

march で検索結果を得られるまでの時間は archie と比べると短い。archie で検索結果が得られるまでに約 18 秒かかったのに対して、march では最長でも約 11 秒で検索結果が得られた。

2.4.2節で述べたように、利用者の探しているファイルはあまり広い範囲を探さなくても見つかることが多いので、大きな `ls-lR` ファイルをもつ anonymous FTP がない、もしくはあっても `ls-lR` ファイルを複数の march サーバで分割して検索しているならば利用者は目的のファイルを素早く発見できることが予想される。

2.4.5 検索時間と通信時間の比較

利用者の体感時間のうち最も長いのが B の検索中の時間帯である。TCP 使用時と UDP 使用時で平均で約 0.5 秒、最大で約 1.7 秒の差がある。これは TCP でコネクションを張るためにいくつかパケットを交換しする必要があるためだと考えられる。しかし検索自体にかかる時間に対してこの時間差は小さく、無視できる。

TCP 使用時と UDP 使用時で所要時間に最も差があるのは C の検索結果の転送の時間帯である。UDP ではただパケットを送り付けるだけであるのに対して、TCP では各パケットに対して到着確認を行なうために遅くなっているものと思われる。しかし B にかかる時間に対してこの時間差は小さく、無視できる。

OS によっては一つのプロセスが保持できるコネクションの数が制限されていることがある。march では広い範囲に検索要求を送信した時に、プロセスの保持できるソケットの最

表 2.2: march 検索、転送の平均所要時間

	A	B	C
TCP で結果を送信 (平均時間)	0.07 秒	2.44 秒	0.46 秒
(最大時間)	0.20 秒	10.84 秒	2.77 秒
UDP で結果を送信 (平均時間)	0.06 秒	2.01 秒	0.07 秒
(最大時間)	0.15 秒	9.17 秒	0.42 秒
TCP と UDP の平均時間の差	0.01 秒	0.44 秒	0.39 秒
最大時間の差	0.05 秒	1.67 秒	2.35 秒

表 2.3: archie の検索+転送の所要時間

A	B+C
0.17 秒	18.05 秒

大数を越える数のサーバが検索結果をクライアントに送信しようとする可能性がある。これを TCP で受信する場合、いくつかのサーバは検索結果を転送できない可能性がある。

UDP はデータグラム型の通信であるため、送信するデータが途中で欠落する可能性がある。しかし anonymous FTP ではミラーによって同じファイルを複数の計算機で提供していることが多い。そのため、検索結果の欠落が起こっても最悪「最も近くにあるファイルが見つからない」という被害で済む。TCP と異なりデータの欠落を防ぐための余分なパケットが回線を流れないので、わずかに速く検索結果を送信できる。クライアントは全てのサーバから送られてきたパケットを一つのソケットで受信するため、OS によるソケットの数の制限は問題にならない。

以上のような理由により、検索結果を送り返すための通信プロトコルとしては UDP が適している。

2.4.6　march サーバの増加による影響

本論文では 8 個の march サーバを広域に分散して実験を行なった。現実に超分散情報を検索するためには march サーバ自身が anonymous FTP と一対一に対応して超分散する必要がある。

march サーバの数が増えた場合の影響としては以下の点が考えられる。

- ユーザの目的のファイルが狭い検索範囲から見つかり易くなる。
- 検索時間は各サーバの検索時間に比例するので変わらない。

- 一度の検索によって多くの march サーバに負荷をかかる。

2.4.7 anonymous FTP 以外のアプリケーションへの応用

超分散情報を検索・提供するシステムで現在インターネットで広く使われているものとしては、World-Wide Web や gopher などの網構造によるシステムが挙げられる。これらの anonymous FTP と異なる点は、同じ情報が複数のサーバで重複して提供されていないことである。

現在の IP マルチキャストは UDP によって配送されるのでパケットの到達保証がないため、これを利用した検索システムでは目的の情報が発見できない可能性がある。

また、同じ情報を複数のサーバで提供していないため、TTL を小さな値から少しずつ大きくするという現在の march の手法は意味を成さない。そのため遠くにあると思われる一つの情報を検索するために日本や世界の全てのサーバが検索を行なう必要があり、現実的ではない。

マルチキャストを利用して World-Wide Web や gopher で提供されている情報を検索するためには、パケットの到達保証のあるマルチキャストの実現が必要であると共に、現在の threshold に代わる、任意の地域やグループをマルチキャストグループとして指定できる手段が必要である。

2.5 まとめ

インターネットではさまざまな情報が提供され、その一部は個人レベルで自由に提供できる。そのため、逆に情報を探す立場に立つと目的の情報の場所が分からない、さらには存在するのかどうかさえ分からないという問題が生じる。本論文ではそのような情報を “超分散情報” と定義し、超分散情報を探す手法について研究を行なった。具体的には、現在の情報検索システムとその問題点について説明した。マルチキャストについて説明し、これが情報検索に有用であることを述べた。 anonymous FTP で提供されているファイルを検索するシステム “march” の実装と評価を行なった。このシステムはサーバ自身を超分散することによって、爆発的に増加するファイルを素早く検索できる。

march は anonymous FTP 上で提供されているファイルを検索するシステム archie と比較して次の長所を持つことを示した。

- データが分散され検索が速い。
- サーバとデータが分散されサーバの負荷が軽い。
- 近くの anonymous FTP 上のファイルを優先的に検索できる。
- 近くから順に探すことにより、遠くのファイルを転送して余計な回線に負荷をかけずに済む。

- anonymous FTP の運用開始/中止に素早く対応できる。
- サーバの激しい増加に対処する必要がない。

しかし現在の実装では以下に示す問題点がある。

- ユーザによる不用意な広域検索を防止できない。
- 検索をする人の居場所によって指定できる検索範囲の柔軟さが変わる。
- パケット落ちによる情報洩れが生じる。

後ろ二つの問題点は現在の IP マルチキャストの実装に起因するものである。パケット落ちによる情報洩れは、ファイルの複製を複数箇所で持ち合う anonymous FTP では大きな問題にならない。しかし World-Wide Web など複製を持ち合わない他の超分散情報を検索する場合には大きな問題となる。

今後の課題としては、任意の地域指定を可能にするための march もしくは MBone の改良、パケット落ちのない MBone の実現が挙げられる。

第 3 章

マルチキャストトランスポートプロトコルに関する提案と考察

本章では我々が提案している信頼性のあるマルチキャストプロトコル SRAMP(Scalable and ReliAble Multicast Protocol) [26] の実装の際に問題になった点と今後の計画を述べる。

3.1 SRAMP の概要

SRAMP は一対多通信において、信頼性のある通信機構を提供するトランスポート層プロトコルである。

マルチキャストはグループ通信を行なうアプリケーションと相性が良い。例えば、メーリングリストやソフトウェアの配布などは、アプリケーション自体がグループ通信の特徴を持っている。そのため、マルチキャストを用いることによってメンバ管理を行なう必要がなくなり、また、トラフィックを削減することができる。

現在、インターネットでは IP マルチキャストを用いたマルチキャストが実現されている。しかしながら、IP マルチキャストはデータグラム型の配送機構であるため、送信したデータが確実に受信されたかどうかの保証がない。メーリングリストやソフトウェア配布を行なう場合、送信したデータが確実に受信者に届かなければならないため、信頼性のある通信を提供する必要がある。

通信の信頼性を確保するためには、送信されたデータの中で受信に失敗したデータの回復を行なう必要がある。受信に失敗したデータの回復を行なう方法としては、次の 2 つが考えられる。

- 受信に失敗したデータを検出し再送する。
- 誤り訂正符号を用いて受信に失敗したデータの復元を行なう。

SRAMP では前者の方法を採用した。受信に失敗したデータを検出すると、送信者はそのデータを受信者に向けて再度送信する。

受信に失敗したデータを検出する方法は一通りだけではない。一連の送信データの欠落部分を検出するための方法としては、次の 2 つが考えられる。

- ACK(Acknowledgment) を用いる方法。
- NACK(Negative acknowledgment) を用いる方法。

前者の ACK を用いる方法は TCP で利用されており、ユニキャストでの信頼性のあるデータ通信に有効であることが実証的に証明されている。しかしながら、マルチキャストで ACK を用いた欠落検出を行なうと次のような問題が生じる。ACK は、送信者が欠落検出を行なう機構である。ユニキャストの場合は受信者が一人だけであるため、送信者は一人の受信者に対してデータの欠落を調べるだけで良い。マルチキャストでは受信者が複数存在する可能性がある。送信者は、どの受信者が欠落を起こしているかを調べなければならないため、すべての受信者について受信状況を把握しておかなければならない。つまり、受信者の数が増えるにしたがって送信者の負担が増大してしまう。

そこで、マルチキャストにおける欠落検出の方法として、多くの場合 NACK が用いられる。NACK は送信された一連のデータに記されている通し番号を調べ、受信者が欠落検出を行なう方法である。欠落を発見した受信者は送信者に再送を要求する。

NACK を用いる場合、欠落検出を完全に受信者に任せてしまうことが可能になる。これにより、受信者の数が増えた場合に ACK で発生した問題を回避することができる。SRAMP も欠落検出のために NACK を用いる。しかしながら、NACK を用いた欠落検出には、次に示すような ACK を用いた場合には見られなかった NACK 特有の問題が存在する。

- 末尾データの欠落検出。
- 送信したデータの保持時間決定。

SRAMP では、末尾データの欠落検出問題を、ダミーデータの送信を行なうことによって解決する。また、送信したデータの保持時間決定は、送信者から受信者までの RTT(Round Trip Time) を基準に決定する。

3.2　SRAMP の問題点と解決案

本節では SRAMP を実装する際に問題となった点を指摘し、解決案の提案と考察を行なう。

3.2.1　SRAMP の実装

現在の実装は、ユーザ空間にプロトコルサーバを置くという形で進められている (図 3.1)。以後、プロトコルサーバを SRAMP サーバと呼ぶことにする。SRAMP を利用する各プロセスは、SRAMP サーバとソケットを用いて接続されている。このソケットを SRAMP ソケットと呼ぶ。SRAMP ソケットの実体は、通常のユニキャストソケットである。

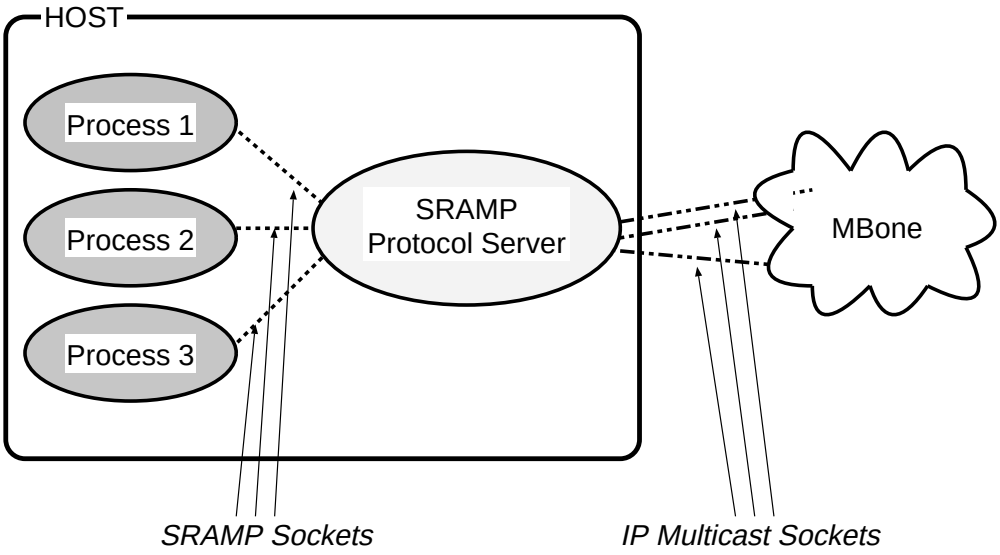

図 3.1: SRAMP Protocol Server

SRAMP サーバは IP マルチキャストソケットを開き、各プロセスが参加しようとしているマルチキャストグループに参加する。そして、プロセスからのデータをマルチキャストグループに送り、また、逆にマルチキャストグループからのデータをプロセスに配送する。

SRAMP プロトコルサーバはデータの送受信を行なう際に、欠落検出、再送処理を行なう。

3.2.2 ソケットのアソシエーションに関する問題

プロセスの識別問題 SRAMP は IP マルチキャストの上位に位置する、トランスポート層のプロトコルである。実際、今回の実装ではネットワーク上でのデータのやりとりに IP マルチキャストソケットを用いている。

通常、マルチキャストを行なうプロセスは、特定のマルチキャストアドレスに関連づけられた IP マルチキャストソケットを開き、そのソケットに対して送受信を行なう。ところが、IP マルチキャストソケットを用いると、各計算機内でソケットから一意にプロセスを識別できなくなるという問題が発生する。つまり、受信者は、どの計算機からデータが送られてきたかは識別できるが、データを送信したプロセスを識別することができない。

この問題の原因はソケットのアソシエーションと、現在の IP マルチキャストの実装に関係する。ソケットは次に示すようなアソシエーションを持っている。

$$\{protocol,\ local\text{-}addr,\ local\text{-}process,\ foreign\text{-}addr,\ foreign\text{-}process\}$$

ユニキャストでは、上記の五つ組の各要素の値が表 3.1左のように設定される。マル

表 3.1: ソケットのアソシエーション

	ユニキャスト	マルチキャスト
protocol	使用しているプロトコル	使用しているプロトコル
local-addr	自分の IP アドレス	マルチキャストアドレス
local-process	自分が使用しているポート番号	使用しているポート番号
foreign-addr	相手の IP アドレス	*local-addr*と同じ
foreign-process	相手が使用しているポート番号	*local-process*と同じ

チキャストの場合、*local-addr*と *foreign-addr*を送受信するマルチキャストアドレスに、*local-process*と *foreign-process*を使用するポート番号に設定する必要がある。同一計算機内では *local-addr*と *local-process*の組でソケットを識別する。そのため、通常同一計算機内では同じ *local-addr*と *local-process*を持つソケットを複数個持つことは許されない。しかしながら、マルチキャストで通信を行なうことを考えた場合、特定のマルチキャストアドレスとポートを用いて通信可能なソケットが、一つの計算機に一つだけしか許されないのは不便である。現行のマルチキャストを用いるアプリケーションは、ソケットに SO_REUSEADDR と、もし提供されていれば SO_REUSEPORT を設定することによって、同一の *local-addr*と *local-process*を複数のソケット (プロセス) に割り当てている。ところが、*local-addr*と *local-process* にマルチキャストアドレスとポート番号を設定するため、例え異なるプロセスから発せられたデータであったとしても、受信側で *foreign-addr*と *foreign-process*をみることによって送信プロセスを識別することができない。

解決案 プロセスの識別問題を解決する方法としては、次の三つが考えられるだろう。

1. 上位層で識別する。
2. 送信と受信に異なるソケットを使う。
3. ソケットの実装を拡張する。

1. の方法は、ソケットレベルで区別するのではなく、データ自信にプロセスを識別する ID を付加する方法である (図 3.2)。もっとも単純な方法で、実現が容易である。欠点としては、上位プロトコルで ID の付加と解釈を行なうため、上位プロトコルに負担がかかることである。

2. は受信用のソケットとは別に送信用のソケットを作り、マルチキャストグループへの送信はすべて送信用ソケットを使う方法である。送信用と受信用のソケットのアソシエーションは表 3.2のようになる。受信は受信用ソケットから行なわれる。受信用

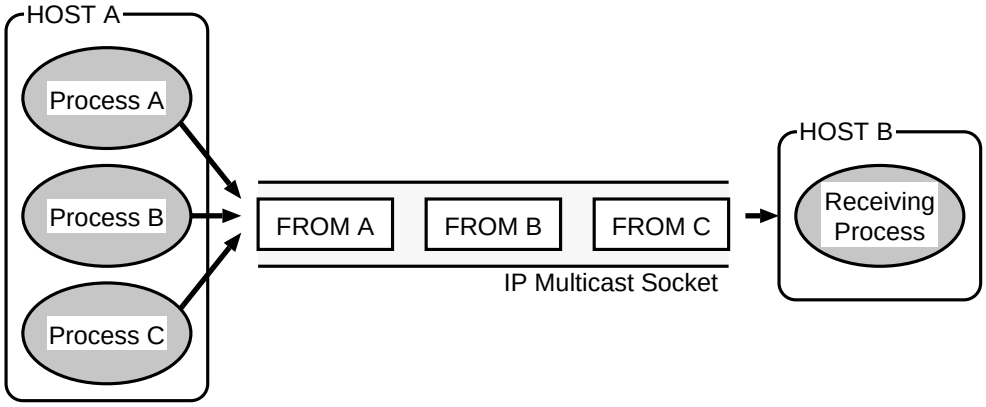

図 3.2: 上位層での識別

ソケットの *local-addr*と *local-process*がマルチキャストアドレスとポート番号に設定されているため、マルチキャストアドレスに送信されたデータを受信することができる。また送信側では、送信用ソケットの *local-addr*と *local-process*を送信者の IP アドレス、使用しているポート番号に設定しているため、受信者は送信されてきたデータの出所がわかる (図 3.3)。

この方式の欠点は、相互通信を行なうためには 2 つのソケットが必要になることである。

表 3.2: 送信、受信ソケットのアソシエーション

	送信用ソケット	受信用ソケット
local-addr	自分の IP アドレス	マルチキャストアドレス
local-process	自分が使用するポート番号	ポート番号
foreign-addr	マルチキャストアドレス	*local-addr*に同じ
foreign-process	ポート番号	*local-process*に同じ

最後の 3. の方法は、IP マルチキャストの拡張を伴う解決法である。2. の方法を拡張したものとも考えられる。この方法で用いるソケットのアソシエーションは 2. で述べた送信用ソケットと同じになる。表 3.2を見て分かる通り、2. の方法では送信用ソケットと受信用ソケットで情報が重複している。同一計算機内のプロセスを識別しながらマルチキャストを行なうために必要な情報は、(送信者の IP アドレス、送信者が使用するポート番号、マルチキャストアドレス、マルチキャストのポート番号) の 4 つで良い。これらの情報は 2. の方法で用いる送信用ソケットに含まれている。この

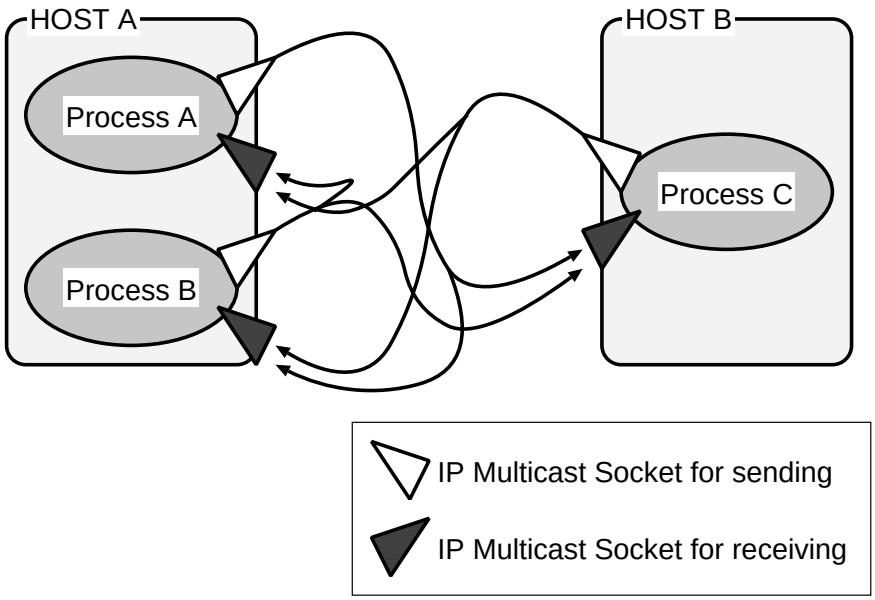

図 3.3: 送受信のソケットを分ける場合

送信用ソケットのみを使ってマルチキャストを行なう。そのためには、IP マルチキャストの受信部分を変更しなければならない。

現在の IP マルチキャストの実装では、受信したパケットが自分宛かどうかを判断するために、パケットの宛先アドレスと宛先ポート番号を、受信に利用するソケットの *local-addr*、*local-process* と比較している。だから、宛先アドレスがマルチキャストアドレスであった場合は、*local-addr*、*local-process*と比較するのではなく、受信に利用するソケットの *foreign-addr*、*foreign-process*と比較するようにすれば良い。

この方法は、1. に見られたプロトコルオーバヘッドや 2. に見られたソケットの浪費の問題がない。

3.2.3　RTT の計測に関する問題

SRAMP は NACK を用いた欠落検出を行なうプロトコルである。3.1節で述べたように、NACK を用いたプロトコルにはパケット保持時間をどのようにして決めたらよいかという問題がある。SRAMP では、送信者から受信者までの RTT を計測し、それを基準に保持時間を決定している。

RTT の変動問題 残念なことに、現在のインターネットでの RTT の計測値にはかなりの変動がみられる。これは IP のデータ転送の性質に原因がある。IP では best effort の

転送方式を採用しているため、トラフィックが集中する箇所で予想以上の転送遅延、あるいは欠落が生じる可能性がある。その結果が RTT の変動となって現れている。

このような状況の中で RTT を計測し、その値を利用すると、現実の状況にそぐわない値を採用してしまうことになる。

解決案 RTT の値が安定しない原因は、IP が best effort の転送を行なうためである。そのため、送信したパケットが目的計算機に到達する時間が不安定になり、結果的に RTT の値が変動する。

そこで、資源予約を用いることを考える。ここでは、資源予約という言葉を、回線の帯域幅とほぼ同等の意味で使う。資源予約を行なうことにより、転送レートが安定しパケット伝送時間の変動が小さくなる。

データリンク層で資源予約を行ない、RTT を安定化させることによって実際のネットワークの状況を反映した通信を行なうことができるだろう。

3.3 今後の展望

本章では我々が提案するトランスポート層マルチキャストプロトコル SRAMP について、実装の際に問題になる点を考察した。

現時点で実装が終了していない部分は、3.2項で述べたトランスポート層でのプロセスの識別と RTT の計測である。トランスポート層でのプロセスの識別は、3.2.2節で提案した IP マルチキャストの拡張によって実現する予定である。RTT の計測に関する問題を本質的に解決するためには、資源予約機構が実現されなければならない。しかしながら、資源予約機構がない場合の問題は、パケット保持時間が実際に必要とされる時間よりも長くなってしまうだけであり、プロトコルが動作しなくなるわけではない。したがって、資源予約を行なわずに実験を進めていってもプロトコルの動作の検証を行なうことができる。

資源予約に関しては、リアルタイムワーキンググループでデータリンク層に ATM を用いた資源予約機構の実現、およびトランスポート層での資源予約プロトコル (Resource reSerVation setup Protocol、RSVP)[27] の実装が予定されている。完成すればそれを用いて資源予約下での動作を評価することになる。

今後さらに実装を進めていき、SRAMP を実証的に評価する予定である。

第 4 章

JP MBone

4.1 JP MBone の現状

MBone は、IP マルチキャストを用いた実験のため Internet 上に仮想的に作られたネットワークであり、当初 1992 年 3 月に San Diego で行なわれた IETF においてその会議の模様を Internet 上に放送するため [28] に始められた。その後も継続的な IP マルチキャストの実験基盤としてメーリングリスト mbone@isi.edu などを通して協調的に実験運営されており、その後の IETF 会議などの中継を重ねるごとに参加組織数、参加国数とも増加している。

一方、日本においてもメーリングリスト mbone-jp@wide.ad.jp が作られ、この参加者による JP MBone 運用グループによって、MBone 構成におけるトンネリングや threshold などの設定調整、JP MBone を利用したマルチキャスト実験の相互調整などが協調的に行なわれている。

現在、JP MBone には 60 を越える組織が参加しており、ここ 1 年間でほぼ倍増している。このため、図 4.1のようにいくつかの JP MBone における拠点を設け、適切になるよう調整しながら各組織をそこに接続することによって実際の IP ネットワークに沿った効率的な接続を協調的に行なっている。

IETF 会議の中継などと同様に、日本国内においてもいくつかの会議やイベントに対して JP MBone を用いて音声や映像などを中継する実験がいくつも行なわれてきており、会場外から JP MBone を通しての会議参加なども実験として行なわれている。ここ 1 年間に行なわれたこれらのリストを表 4.1および表 4.2に示す。

4.2 JP MBone の運用と諸問題

4.2.1 threshold と初期 TTL

現在の IP マルチキャストの技術は発展途上にあり、また、現在はまだ MBone 上のすべてのマルチキャストルータが真のマルチキャストや流量制限をきちんと実現できているわけではない。そのため、音声や画像などでの多量のマルチキャストパケットの送出は、MBone

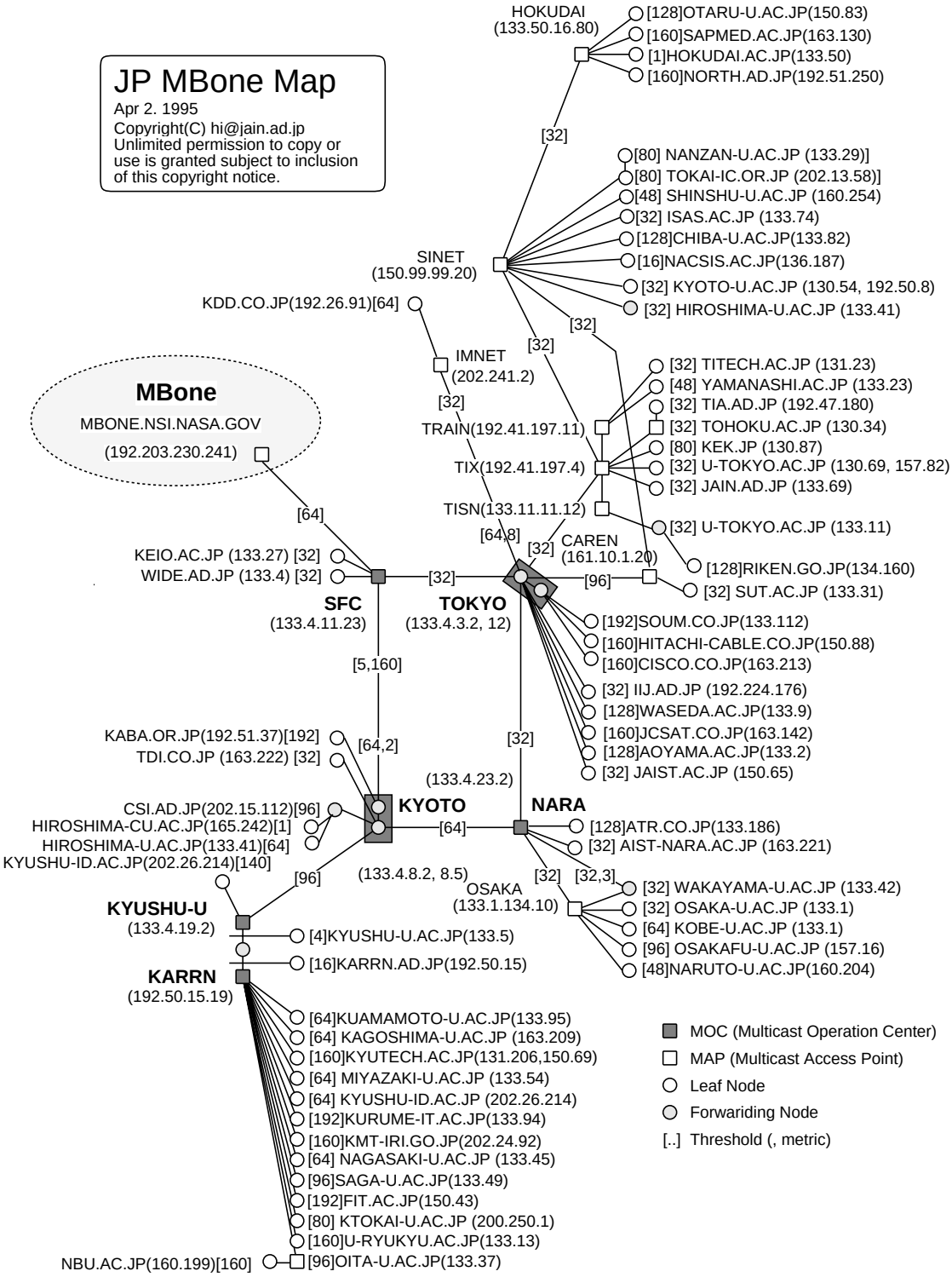

図 4.1: 国内の MBone 接続トポロジー図

表 4.1: JP MBone 利用で行なわれた会議・イベント一覧

日付	内容
94 年 4 月 22 日	第 1 回京都大学高度情報化フォーラム 京都大学医学部附属病院 (vat nv)
94 年 5 月 18 日〜19 日	大阪大学総合情報通信網 (ODINS) の完成披露式典 大阪大学 (vat nv wb)
94 年 6 月 1 日〜3 日	JAIN Consortium 第 3 回 総会・研究会 KKR 甲府ニュー芙蓉 (vat nv wb)
94 年 6 月 20 日	九州大学キャンパスネットワーク KITE 完成披露式 九州大学 (vat nv wb)
94 年 6 月 24 日	九州工業大学マイクロ化総合技術センター開所記念 九州工業大学マイクロ化総合技術センター (vat nv)
94 年 6 月 24 日	奈良先端大見学会 奈良公園 (vat nv)
94 年 6 月 24 日	94 年度 KARRN 総会 九州大学大型計算機センター (vat nv)
94 年 7 月 13 日〜14 日	グローバルクラスルームプロジェクト 佐賀大学・武雄北中学・大和中学校 (vat nv wb)
94 年 7 月 27 日〜29 日	ITE テレビジョン学会 '94 佐賀大学 (vat, nv, wb)
94 年 9 月 10 日	CSI フォーラム広島 広島県健康福祉センター (vat nv)
94 年 9 月 24 日	電気関連学会九州支部連合大会 九州東海大学 (vat nv wb)
94 年 9 月 26 日	電子情報通信学会 平成６年秋期全国大会特別企画 東北大学川内キャンパス (vat)
94 年 10 月 7 日	IP Meeting '94 東京工業大学大岡山キャンパス (vat nv wb)
94 年 10 月 25 日〜27 日	JAIN Consortium 第 4 回 総会・研究会 志賀の島国民休暇村 (vat nv wb)
94 年 11 月 6 日	大阪大学エンターテイメントイベント 大阪大学豊中キャンパス (vat nv)
94 年 11 月 12 日	日本科学教育学会九州支部発足記念研究大会 佐賀大学 (vat nv wb)

表 4.2: JP MBone 利用で行なわれた会議・イベント一覧 (続き)

日付	内容
94 年 11 月 14 日〜15 日	日本マイクログラビティ応用学会第 10 回学術講演会 大阪工業技術研究所 (vat nv wb)
94 年 11 月 16 日	Donald B. SIMPSON 氏 講演会 福岡アメリカンセンター 2F ホール (vat nv wb)
94 年 11 月 22 日	KARRN シンポジウム 1994(飯塚) 九州工業大学飯塚キャンパス (vat nv wb)
94 年 11 月 21 日〜22 日	第 2 回 人工物工学シンポジウム 東京大学山上会館 (vat nv wb)
94 年 11 月 25 日	電子情報通信学会専門講習会 大分県ソフトパークより ISDN で大分大学 (vat nv)
94 年 12 月 1 日〜2 日	平成 6 年度情報処理教育研究集会 九州大学記念講堂 (vat nv wb)
94 年 12 月 15 日	第 2 回 JAIN Consortium シンポジウム 千里ライフサイエンスセンタービル (vat vic wb)
95 年 1 月 20 日	CSI インターネットシンポジウム '95 in 高知 高知県民文化ホールグリーンホール (vat vic wb)
95 年 2 月 10 日	教育におけるインターネット活用実験の現状と課題 宮城教育大学 (vat wb)
95 年 2 月 10 日	東北大学 SuperTAINS デモンストレーション 東北大学 (vat vic wb)
95 年 3 月 7 日	京都大学大型計算機センター第 44 回研究セミナー 京都大学大型計算機センター (vat vic wb)
95 年 3 月 17 日	KARRN 協会講演会 (宮崎) 宮崎厚生年金会館 (vat vic wb)
95 年 4 月 3 日	日本解剖学会 100 周年ミニシンポジウム'95 東京大学医学図書館大会議室 (vat vic)
95 年 4 月 6 日〜7 日	第 1 回 国連大学「ゼロ・エミッション会議」 国連大学 (vat nv)
95 年 4 月 11 日	京都大学入学式 京都大学 (vat nv wb)

全体へ影響を与えることがあり、慎重な設定運用と調整を必要とする。

本来、MBone のリンクに設定される threshold の値は組織内外や地域内外といった領域を設定するために用いられる。例えばある地域の境界上では threshold を 64 以上という値に設定することによって、その地域内から初期 TTL を 63 以下で送出することにより、その地域内だけに届くことを容易に実現できるようにしている。

しかし、現在は完全なマルチキャストが実現できていないため、当面の処置として、threshold の値をそのリンクで MBone として利用可能なバンド幅に応じて設定し、初期 TTL を送出バンド幅に応じて変えることによって細いリンクへ大きなトラフィックが流れないように運用している。

JP MBone においては当初、MBone で利用している国内リンク及び海外リンクが細かったなどの事情に合わせて独自の設定を定めて運用していたが、JP MBone の海外リンクとして利用している WIDE の海外リンクが 1.5Mbps へと増速されたことなどにより、IETF での放送基準などに基づき新しい国内での取り決めによって運用を始めている (表 4.3)。

表 4.3: 国内での利用バンド幅別 threshold 設定

使用される上限目安	→	threshold
制限なし	→	1 (自由)
768Kbps	→	32 (組織境界相当)
512Kbps	→	64 (IETF videocast x 2 channel 相当)
384Kbps	→	96 (IETF videocast x 1 channel 相当)
192Kbps	→	128 (IETF PCM x 2 channel 相当)
96Kbps	→	160 (IETF PCM x 1 channel 相当)
32Kbps	→	192 (IETF GSM x 2 channel 相当)
16Kbps	→	224 (IETF GSM x 1 channel 相当)

最大どれだけのバンド幅を MBone で使用されてよいかは、その物理的なリンクの管理者、あるいはネットワークプロジェクトの担当者と相談の上決定し、それに基づいて threshold の値を設定する。例えば物理的なリンクのバンド幅にかかわらず、MBone で使用する上限目安を 512Kbps にしたい場合は、表 4.3より threshold=64 とする。

一方、イベント中継など音声や映像などで大きなトラフィックが送出される場合は、同じ時間に行なわれる他の利用者とも事前に十分調整した上で、その利用総量の上限目安に応じてそれぞれの送出の初期 TTL を決定する (表 4.4)。

4.2.2 boundary

以前の JP MBone では海外リンクが細いなどの事情で threshold が大きく設定され、それと同時に初期 TTL=127 で送出すると国内全域に届くという設定運用がなされていた。

表 4.4: 国内での利用バンド幅別初期 TTL 設定

使用される上限目安	→	初期 TTL
組織内	→	31 以下
768Kbps まで	→	63 以下
512Kbps まで	→	95 以下 (IETF videocast x 2 channel 相当)
384Kbps まで	→	127 以下 (IETF videocast x 1 channel 相当)
192Kbps まで	→	159 以下 (IETF PCM x 2 channel 相当)
96Kbps まで	→	191 以下 (IETF PCM x 1 channel 相当)
32Kbps まで	→	223 以下 (IETF GSM x 2 channel 相当)
16Kbps まで	→	255 以下 (IETF GSM x 1 channel 相当)

ところが今回のバンド幅と threshold の関係の大幅な設定変更にともない、国内の細いリンクを持つところへも届くように送出すると、当然ながらより太い海外リンクで設定している threshold も越すことになり、初期 TTL の値だけでは国内だけへ届ける放送などを実現できなくなった。

一方、新しい IP マルチキャストカーネルにおいては boundary という概念が導入され、239.0.0.0/8 の領域の一部に対して、MBone のそれぞれのリンクに境界設定ができるようになった。これを利用して、JP MBone では海外リンクに対して 239.133.0.0/16 という boundary を設定し、JP MBone 内で 239.133.0.0/16 の領域を用いた場合はその TTL などに関わらず国内のみだけに制限して届かせることができるようになった。

したがって、現在の JP MBone の運用においては、国内のみで放送するセッションなどは必ず 239.133..xxx.yyy を用いることとし、利用バンド幅に応じて上述の表を目安に初期 TTL を決定する。一方、239.133.xxx.yyy 以外のアドレスを使うことによって海外へもパケットを流す利用の際には `mbone@isi.edu` などへ通知を行ない調整する。

4.2.3 sd と sd.jp

MBone においては、vat や nv などを用いて会議やイベント中継などは、一般に sd (session directory) というツールを用いてそのセッション情報が MBone 上へアナウンスされる。現在の sd は SDP (session directory protocol) version 1 にしたがい、そのセッションの初期 TTL と同じ初期 TTL を用いて、セッションについての各種情報を 224.2.127.255 のグループへアナウンスしている。

一方、国内向けのセッションを 239.133.xxx.yyy にて例えば TTL=191 で行なおうとすると、sd を用いた場合、そのセッション情報が 224.2.127.255 へ TTL=223 にてアナウンスされることになり、結果として、実際には国内にしか放送されないセッションについての情報が世界中へとアナウンスされるという不自然なことが起きてしまう。

この問題を解決するため、国内向けのセッション (239.133.xxx.yyy を利用) のアナウンスをする時には 224.2.127.255 でなく 239.133.127.255 を用いてアナウンスを行なうことにし、この 239.133.127.255 で送受信する sd を作成して、従来の world wide での sd と区別して、sd.jp と呼んでいる。

現在、sd 自体のソースが公開されておらず、また、アナウンスのためのアドレスを指定する手段も用意 (公開) されていないため、現時点ではバイナリパッチにより sd.jp を自動作成して用いており、ftp://ftp.kyoto.wide.ad.jp/multicast/sd.jp/ などから入手することができる。

実際の利用にあたっては、従来の sd の代わりに sd.jp を立ち上げれば国内向けセッションについての情報を見ることができるが、このままでは world wide にアナウンスされているセッションについての情報を同時に見ることができないため不便である。そこで、224.2.127.255 へのすべてのアナウンスを 239.133.127.255 へと中継する sd_forwarder というプログラムを動かすことによって、この問題を解決している。つまり、現在の運用においては、sd.jp を動かすだけで国内向けセッションも world wide 向けのセッションも一般ユーザは区別することなく受信・表示・利用することができる。

4.2.4　トンネルの迂回問題

マルチキャストをサポートしていないルータが多く存在する中、MBone においてはマルチキャストをサポートするホスト同士をトンネルで結ぶことによってリンクを形成している。このトンネルは mrouted.conf において自分 (のインターフェース) と相手を指定することによって設定する。

バックボーンなどでループ構造がある場合や、マルチホームの組織の場合、トンネルの両端のホストの間に複数の経路が存在し得るため、通常時の経路に沿って効率よくトンネルを設定したつもりでも、種々の原因で別の経路に変わった時に、結果的にトンネルが意図しない経路を通ってしまうことがよくある。特に、(ユニキャスト的に) マルチホームの組織がバックボーン A とバックボーン B へリンクを持っている時、MBone におけるトンネルをバックボーン A にあるホストとの間で設定していたとすると、もしも何らかの障害などでその組織とバックボーン A とが切断された場合、トンネルを設定している相手ホストとの経路がバックボーン B を通って大きく迂回してしまうことになる。

このような MBone トンネルの意図しない迂回は、迂回経路上に多くのトラフィックを生じさせる可能性があり、避けることが望ましい。このためには、例えば、次の 2 つの方法で避けることが可能である。

- SSRR(Strict Source and Record Route) などで相手ホストまでソースルーティングを行なう
- 迂回経路側のルーターでトンネルパケットのフィルタリングを行なう

ここで、LSRR(Loose Source and Record Route) ではなく SSRR である必要があるのは、途中のリンクだけがなんらかの障害で落ちてた場合、LSRR だと、迂回ルート経由になる可能性があるためである。しかしながら、SSRR については、パケットが大きくなるなど効率が悪い、SSRR を使えないもしくは使わせない設定のルータがある、サポートするには mrouted や kernel の大幅な変更が必要、といった問題がある。また、フィルタリングを行なう場合も、トンネルと関係ない側のルータで設定しなければならない、CISCO などプロトコル別にうまくフィルタリングできないルータが多く、トンネルで使われている IP カプセル用のプロトコルだけを指定してフィルタリングできない、など、現実には適用することができなかった。

そこで、発想を変え、意図しない迂回ルートになってしまった場合に、ホストが相手ホストからの MBone トンネル設定が認識できないようにし、迂回した MBone トンネルが張られないようになるという方法を実現した。実際には非常に単純な実装で、mrouted 同士が交換する IGMP の DVMRP の PROBE/REPORT パケットのみを指定した低い TTL で送出することによって、迂回ルートを通ってしまうと PROBE パケットが届かずに迂回トンネルが起こるのを防いでいる。

具体的には、mrouted.conf の tunnel の記述において、

```
tunnel <local-addr> <remote-addr> [srcrt] [ttl <l>] [metric <m>]
                                  [threshold <t>] [rate_limit <b>]
```

と、TTL が指定できるように mrouted を拡張した。これにより、実際の運用ではトンネルの相手ホストまでの通常のルートにおけるホップ数を TTL として指定すれば、多くの意図しない迂回トンネル問題は防げるものと思われる。WIDE バックボーンにおいてもループ構造を持っているため、一部においてこの拡張された mrouted を用いて運用しつつある。これらは ftp://ftp.kyoto.wide.ad.jp/multicast/mrouted/ などから入手できる。

4.2.5 ポリシールーティング

現在 MBone で主に使われている DVMRP は本来 IGP であるため、その適用領域はあまり大きくとれないとともに細かいポリシールーティングを行なうこともできない。ただし、リンクに対して metric を指定することができ、これによって MBone 網のリンクの重み付けをすることで、ある送出ホストを根とする木の形 (経路) にポリシーを反映させることができる。

例えば、MBone 網の一部に $A-B-C-D-A$ というループ構造があった場合、$D-A$ 間にだけ metric=4 を与えて他をすべて metric=1 にすれば、$A-B-C-D$ という経路は合計 metric=3 のため、MBone 網における木において $D-A$ 間を通るものはなくなる。そして、なんらかの障害で $A-B-C-D$ の経路の一部が切れてなくなれば、代替経路が他になければ $D-A$ 間を通る木が作られることになる。これを、MBone におけるバックアップリンクと呼ぶ。一般には $A_0-A_1-\cdots-A_n$ に対して $A_{i-1}-A_i$ の metric 値を

m_i とすると、$A_0 - A_n$ 間に $\sum_{i=1}^{n} m_i + 1$ 以上の metric 値でリンクを張れば、バックアップリンクとなる。そのリンクは、実際に対応する物理リンクが存在している場合もあれば、$A_0 - \cdots - A_n$ 上の仮想的なリンクである場合もある。

一方、おなじく $A_0 - A_n$ に metric 値 $\sum_{i=1}^{n} m_i - 1$ にて $A_0 - \cdots - A_n$ 上の仮想的なリンクを張ると、おもしろいことが起こる。つまり、物理的なリンクにおいては何らパケットが送られる点に変わりはないが、MBone 上において A_0 と A_n の間の通信のみ、$A_0 - A_1 - \cdots - A_n$ のリンクではなく、$A_0 - A_n$ という仮想リンクの方を通るようになる。これと、thershold の設定をうまく組み合わせると、他へは伝播されないプライベートな通信を行なうことができる。例えば、$A_0 - A_1$ および $A_{n-1} - A_n$ のリンクの threshold を 32 としたとき、$A_0 - A_n$ の threshold を 16 とすると、A_0 と A_n 双方で初期 TTL=31 にて送出すれば A_1 や A_{n-1} へはパケットが送られないため、A_0 と A_n のプライベートリンクが実現する。ただし、現実の MBone の運用においてはこのような設定は一つ間違うと混乱を招いて他への影響が大きいため、実際には行なわないようにしている。

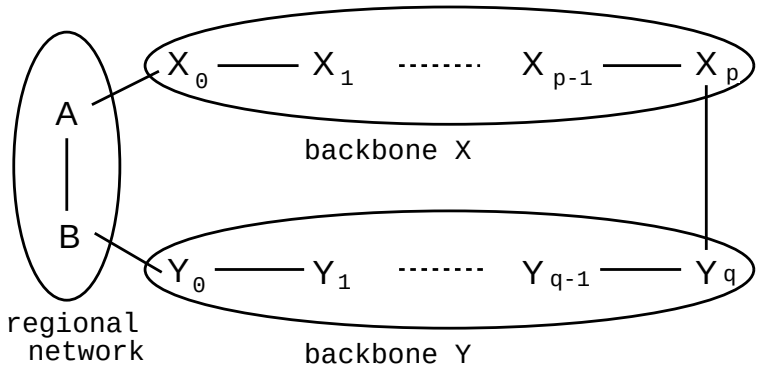

図 4.2: 地域ネットが二つのバックボーンにつながっている場合

この他にも、JP MBone 上では少し複雑なポリシーを反映させたいことがしばしば現実のケースとして出てきている。ある地域ネットが $X_0 - X_1 - \cdots - X_p$ を持つバックボーン X と、$Y_0 - Y_1 - \cdots - Y_q$ を持つバックボーン Y の双方に接続しており、その地域ネットのホスト A は X_0とリンクを持ち、もうひとつのホスト Bは Y_0とリンクを持ち、もちろん、$A - B$ と内部でリンクを持つ一方、X と Y は $X_p - Y_q$というリンクを持っているとする (図 4.2)。これは、大きなループ $A - B - Y_0 - \cdots - Y_q - X_p - \cdots - X_0 - A$ を持つわけであるが、実際に対応する物理リンクをそれぞれ持つため、なんらポリシーがなければすべての metric 値は 1 でも問題はない。しかし、X と Y は、相互の通信を Xp と Yqの間だけでするポリシーを持ってユニキャスト的にはそのように通信が行なわれている場合を考える。X_0 と Y_0の間の通信はユニキャスト的には $X_p - Y_q$を通るようにされているとしても、マルチキャスト的には $X_0 - A - B - Y_0$ のほうが $X_0 - \cdots - X_p - Y_q - \cdots - Y_0$ よりも合計 metric 値が小さいことは一般に考えられ、現実にもこのケースが発生した。単純にバックアップ

リンクの手法を用いて $A-B$間の metric 値を十分大きくすると解決するように見えるが、地域ネット側のポリシーとして A と Bのトラフィック自体は $A-B$間を通って欲しいというのが満たされなくなってしまい、一般にはそれでは解決しない。ただし、これらのポリシーだけを満たすだけでよいならば、この場合は、threshold 設定を除いた上述のプライベートリンクの手法、つまり、$A-B$間の metric 値を $A-X_0-\cdots-X_p-Y_q-\cdots-Y_0-B$ の合計 metric 値より 1 だけ小さい値にする、を用いれば、地域ネットと X と Y すべてのポリシーを満たすことができる。

現実には、metric 値の調整をするにしても複雑になり過ぎることが多く、また、このような metric 値の調整だけではすまない難しいポリシーを持つ場合もあり、種々のポリシーをうまく反映できるマルチキャストルーティングプロトコルが望まれる。