第11部

信頼性を有するマルチキャスト通信技術

第 1 章　はじめに

本章では，Reliable Multicast WG の最初の 1 年間の成果について述べる．

1.1 背景

昨年の報告書でも述べたが，Reliable Multicast (RM) 技術の必要性は非常に高まっている．IETF では，Reliable Multicast Transport (RMT) WG が発足され，一部の技術は研究段階のものからビジネスで利用できる標準化が求められるレベルに達している．しかし当然まだ輻輳制御など非常に困難な open research issue も多数あり，大規模な実験・運用という課題も残っている．WIDE プロジェクトでは，live network で最新技術を導入した大規模な実験が可能なので，既存の技術，本 WG から生まれる技術を実際に動かしていくことを目標にして RM WG は活動している．

活動をはじめて 1 年経て，新たなプロトコルの提案，RM をサポートするための framework 実装などを成果として報告できるものがでてきた．本章ではこれらについて紹介する．

1.2 IETF の動き

去年の報告書では，99 年 3 月の IETF のミーティングで RMT BoF が行われ，それについて報告した．それから，1999 年 7 月の IETF では，正式に WG として集まり，その後開催された 1999 年度の全ての IETF でも，RMT WG としてのミーティングは行われた．

この間ドラフトの数も増え，チャーターも 1 回書き直され，WG の目的・方針も明確になった．具体的には，RM プロトコルを種々に渡って検討していくため，3 つのプロトコル例 protocol instantiation (PI) を標準化する．またこの 3 つの PI をサポートするために必要な building block (BB) の標準化も行う．

1.2.1 Protocol Instantiations

現在，標準化される PI は以下のものである．

- NACK-based Protocol
- Tree-based ACK (TRACK) Protocol
- Open Loop Protocol that uses Forward Error Correction
  これは，ALC (Asynchronous Layered Coding) と呼ばれている

1.2.2 Building Blocks

現在提案されている BB には以下のものがある．

- 自動木構成 BB (現在ドラフトは公開されてない)
- NACK Oriented RM (NORM) BB (現在ドラフトは公開されてないが存在する)
- ACK BB (現在ドラフトは公開されてないが，ML ではメモみたいなものがまわされている)
- FEC BB
- Generic Router Assist (GRA) BB

今後更に細かく機能が分かれる可能性もあるし，BB も増えることも考えられる．

特に，まだ輻輳制御 BB の標準化が全く進められていない．IRTF のほうで，輻輳制御 (CC) の研究が当初予定されていたペースで進行してないのが主な原因である．IETF では，まだ unicast 用の CC も確定されてなく multicast 用はまだまだ研究段階であり，当分先になると思われる．しかし現時点でも CC に関する情報は，一部 web で公開されている．TCP Friendly Web Page には，さまざまな情報へのポインタが収集されている．(cf. http://www.psc.edu/networking/tcp_friendly.html)
また，RMTP-II を開発した Talarian の web page では，requirements draft や Strawman specification が置いてある．
(cf. http://www.talarian.com/rmtp-ii/)

また，IRTF の SMuG の方では，セキュリティ機能をまとめた BB を提案している (cf. draft-irtf-smug-gkmbb-gsadef-00.txt)

1.3 その他動向

標準化の動きだけではなく，リサーチおよびビジネスの世界でも RM に関する関心は高まっている．相変わらず Talarian は RMTP-II を商品としてプッシュしている．その他アプリケーションと組み合わせた RM パッケージの商品化の動きなども見える．(cf. http://www.whitebarn.com/) Whitebarn は PGM をベースとしたマルチキャストファイル転送アプリケーションを商品化している．(執筆中 Talarian に買収されたことが判明した)

1.4 WIDE RM WG の活動

WIDE RM WG の活動については，主に WIDE 標準パスワードで http://www.shio.org/rm/ にまとめてある．

1999 年度においては，合宿でのミーティング以外に，7 月に UCB からのサマーインターンの学生の発表を含んだ小さなワークショップを行った．また，夏のボード合宿ではテーマであったマルチキャストにあわせて，いくつかのプロトコルの tutorial なども行った．

1.5 報告書の内容

活動を開始し，プロトタイプ実装，既存システムの応用評価と，かなりの成果がみえてきた．今回はそれらについて報告する．まず，プロトタイプ実装としては，FEC を導入した IPv6 の拡張について述べる．次に拡張プロトコル技術として WG からの二つの提案について解説する．一つ目は，優先度を考慮した FEC 機構についての検討である．二つ目は，セキュリティと RM を統合した新しいプロトコルの設計・実装・評価についてである．この技術は適応性の高い RM の研究の第一歩だと考えている．最後に，既存の PGM の実装を WIDE で開発され，使用している DVTS アプリケーションへ応用し，評価した実験について述べる．

今後も上記の拡張や新たな拡張プロトコル技術を確立していく．さらに，他の WIDE WG や，アプリケーション技術と RM 技術を組み合わせて行くことを目指す．最後に，広域で実験・運用するという大きな課題がある．

第 2 章　Reliable Multicast − Using FEC on IPv6

2.1 概要

現在のマルチキャスト通信は、IP マルチキャストをベースに UDP を使用しており、リライアブル（信頼性のある）なデータ通信は提供していない。データ通信には誤りのないパケット転送を実現する方法として、パケット再送による Feedback の方法と、受信側で訂正を行なう前方誤り訂正（FEC:Foward Error Correction）制御による Open Loop の方法がある。現在のような物理的に巨大化したネットワークで、受信者が多い大規模なマルチキャスト通信を行なうためには、パケットを再送する方法ではオーバーヘッドが非常に大きくなるが、FEC はトラヒックの増加を抑える有効な方法であると考えられている。

本研究では、この FEC に着目し、リライアブルマルチキャストを実現する一方法として実装・評価を行なっていく。

2.2 現状

IPv6 で動作する FEC を、IPv6 フラグメントヘッダに、誤り訂正コード生成および訂正コードを用いて IP データグラムを復元する拡張機能を持たせた実装を行ない、ユニキャストでの性能評価を得ている。

2.2.1 仕様

送信側では、まず IP データグラムに対して誤り訂正コードを生成し、IPv6 フラグメント機能を利用して IP データグラムと誤り訂正コードパケットをセグメント化して送信する。

受信側では、セグメント化されたパケットをもとの IP データグラムにリアセンブルする。パケット廃棄があった場合には誤り訂正コードを用いて復元する機能を持たせている。

IP データグラムをセグメント化したものがパケット廃棄なくすべて揃えば、誤り訂正コードを用いて復元する必要はない。したがって、復元せずに再生のみ行ない、訂正コードパケットは廃棄する。この機能を実現するために、IPv6 フラグメントヘッダの

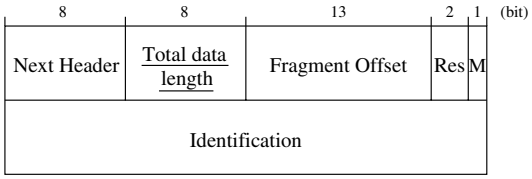

図 2.1 IP version6 Fragment Header

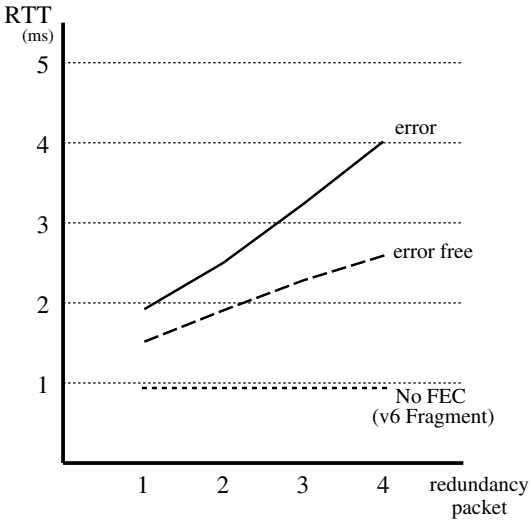

図 2.2 Round-Trip-Time の計測

予約フィールドに、IP データグラムのセグメント化したパケット総数（Total data length）を記述する（図 2.1）。Total data length は 8 オクテット単位の 8 ビット無符合整数で表現する。

誤り訂正符合は RS 符合 ($G(x) = x^8 + x^4 + x^3 + x^2 + 1$) で、パケット廃棄のみを訂正する。

2.2.2 ユニキャストでの性能評価

同一リンク 2 台の FEC 実装ホストで RTT を計測することにより性能評価を行なった。FEC は FreeBSD 3.4R のカーネル内の IP 層で実装し、IPv6 には kame を用いている。ハードウェアには Pentium II400MHz の CPU を持つ PC、メインメモリは 128MB を用いた。IP パケットの分割サイズは 512 byte とし、データ数 8 個に対し、訂正コードを 1〜4 個生成して計測を行なった（図 2.2）。パケットロスはパルス的なものとし、送信時にカーネル内で落した。

横軸を訂正コードパケット数、縦軸が RTT とする。NOFEC が FEC 拡張を行なっていない IPv6 Fragment の RTT を示す。error free は訂正コードパケットを生成するがパケットロスは発生させない（FEC 復元は行なわない）。つまり訂正コード生成の処理時間を示す。error は訂正コードパケットを生成しパケットロスも発生させる（FEC 復元を行なう）。つまり訂正コード生成および FEC 復元の合計処理時間を示す。本実装では、一つのパケットを生成する処理時間は 0.17ms、復元する処理時間は 0.35ms であることがわかった。

2.3 今後の展望

まず、実装した FEC のマルチキャストでの性能評価を行なう。つぎに、全ての送信データに FEC を行なうのではなく、データを階層化することにより FEC を行なう箇所・行なわない箇所に分けて、消費帯域を少なくすることを考える。そのためには、動画像に何を用いるかについても考察しなければならない。

第 3 章　傾斜優先度をもつ FEC

FEC (前方誤り訂正) は、送信側で送信データに誤り訂正符号を付加して送出し、転送中に発生するデータの誤りを受信側で訂正する方式である。本研究では、特に IP ネットワークにおいて輻輳などの理由によりパケットが紛失した場合の回復に用いる技術について検討する。

FEC によるパケット紛失回復の技術はすでに詳しく検討されている。具体的には、磁気記録媒体上のビット誤りや電波通信路上で発生するビット誤りを訂正するために用いられている誤り訂正符号による訂正に加えて、パケット紛失がバースト誤りであることに対する対策としてのインターリーブ技術の利用、紛失位置が特定できる場合に訂正効率が向上する消失誤り回復技術の利用、符号系として効率のよい Reed Solomon 符号の利用などが提案されている。

FEC はパケット紛失率の変動に対して有効な手段をもたない。回復できる紛失率上限は送信時に与える冗長度によって決定されるからであり、それを超える誤り、紛失が発生すると、訂正回復することが出来ない。輻輳状態に合わせて送信側で冗長度を適応的に変化させる手法が考えられるが、一般には輻輳状態の測定、送信側への通告、符号生成の 3 段階に要する時間は縮めがたく、その期間はパケット紛

失が発生しつづける。

そこで本研究では、転送データが階層的エンコーディングを施されたマルチデータである場合、その優先度により異なる冗長度を与える FEC を提案する。

ここで前提となる階層的エンコーディングとは、データを異なる優先度をつけた部分に分解できる、マルチメディアのコーディング手法であり、たとえば画像の場合、粗い部分のデータを高優先度で、それに加えて細かい部分を再生するためのデータを低優先度で転送することを考える。帯域の都合でデータの一部しか転送できない場合、高優先度の粗い画像のみを送り受信側で再生する。もし帯域が十分あれば低優先度の細かい画像への追加データをあわせて送り、受信側では粗いデータと細かい追加データを併せて再生することにより、きれいな画像を表示できる。このようなエンコーディングは、静止画の場合は空間解像度 (画像の粗さ細かさ)、色調解像度 (色合いの粗さ細かさ) の 2 つの次元に関して階層を持たせることができ、動画の場合にはそれに加えて時間解像度 (動きの粗さ細かさ) の次元に関して階層を持たせることが出来る。通常は空間解像度と時間解像度について試みられることが多く、空間解像度の階層化は Wavelet 変換などを用いると自然に実現できるが、JPEG や MPEG のマクロブロックによる方式では導入が難しい。時間解像度は単にフレームを抜くという操作で実現でき、MPEG の I フレーム・B フレームの違いを用いた階層化を主張する研究もあるが、MPEG の B フレームはデータ量が少ないので階層間のデータ量比率が希望するものにならないことが多い。

また、マルチメディア、つまり複数のメディアデータが複合している場合、たとえば動画と音声の複合や、さらに動画、音声、テレテキストの複合などの場合は、メディア間で優先度に格差をつけることが可能である。動画に対して音声に高い優先度を与えると、非輻輳時には音声と動画が、輻輳が激しくなると音声のみが継続再生され、輻輳が軽減するとまた音声と動画が再生される、ということが可能になる。これにより、視聴者は中断無く再生を見ることが出来るので、動画と音声が同時に中断してしまう場合に比較してストレスが小さいものと予想される。同様に、テレテキスト部分に最も高い優先度を与えると、服装の激しいときにはテキストだけが継続再生するということが可能になる。

上記のような階層的エンコーディングを施したデータが与えられたとき、FEC により優先度の高いデータをより多く回復できるように、冗長度のより大きい冗長符号を付加する方式を提案する。たとえば優先度の異なる H, M, L の 3 層からなるデータを転送する場合、優先度の高い H 層のデータに対して 25%、中間の M 層のデータに対して 10%の冗長度を加え、優先度の低い L 層のデータに対しては冗長度を加えないとする。パケット紛失が無い場合は H, M, L のすべての層のデータが正しく受信でき、再生される。輻輳が起こり 10%以下のパケット紛失が発生するときは、L 層のデータは復元できないが、H, M 層のデータは紛失を回復できる。輻輳が激しくパケット紛失が 10-25%になると、H 層だけが回復でき、M, L 層は回復できなくなるが、それでもサービスは継続できる。紛失率が 25%を超えるとすべて回復できなくなる。

この方式の特徴は、輻輳状況が変化して紛失率が変動する環境でも、送信側に何も操作を加えなくても紛失率レベルに合わせたレベルのデータを中断無く再生できるということである。

この方式の有効性を検証するために、冗長度の組み合わせを設計すると同時に、パケット紛失の実測データを元にして冗長度の組み合わせが有効に働くかどうかを求めた。

初めに、冗長度の組み合わせの例を提示する。実働環境の例として、比較的狭帯域の TV 電話を想定する。具体的には 16Kbps の音声データ、64Kbps の動画データがあるものとし、さらに動画が粗い部分の 32Kbps と細かいところを補う部分の 32Kbps の２つに分割されるような、3 階層の階層型エンコーディングが可能であるとする。また、FEC を導入したために起こる転送遅延を最大 50mS までにとどめることを条件とする。これは対話が成り立つための遅延上限である 300mS 程度から考えて仮定した値である。この遅延の上限によって、インターリーブブロックへ滞留できるデータ量が 50mS 分に限定され、それは 16Kbps 音声では 100B にあたる。

この 100B を 4 つのデータパケットに分割し、それに冗長度を付加することとする。この音声を高優先度の H 層とし、これに対しては 4 データに対して 1 データの冗長度を加えることにする。その結果、パケットの送出間隔は 10mS (50mS に 5 パケット) となり、ここのパケットには 25B のデータを持つ。ま

た、動画データは 32Kbps の 2 層に分解するが、粗い画面を与える部分を中優先度の M 層とし、音声より少ない冗長度を与え、細かい画面を与える部分を低優先度の L 相とし、冗長度なしとする。M 層の冗長度の与え方は後述する。

個々のパケットは 25B であるので、このままではパケットヘッダー（IP、UDP、RTP ヘッダーを合わせて 42B）のオーバーヘッドが大きくなる。対策として、動画データを別パケットとするのではなく、同一パケット内に入れ込む。ここで、同一パケットに入れ込むことが必ずしも M、L 層の動画データに対して同じインターリーブブロック長を使うことを意味しない。パケットの送出時間間隔は同一であるが、インターリーブ長を層によって変えることは、可能である。ここの場合でも、H 層の音声に対しては遅延の上限を 50mS とするためにインターリーブブロック長を 100B、その中を 4 分割して 1 冗長データを追加し全体を 5 パケットとすることとしたが、M 層に対してはインターリーブブロック長を 18 パケットとし、それに対して 2 冗長データを追加し全体を 20 パケットとする。パケットの送出間隔を 10mS としたので、20 パケットブロックは 200mS に相当し、32Kbps の発生率の動画データであるから 800B 分に相当する。この 800B 分のデータを 18 パケットに分割すると 45B となり、それに 2 パケット分の冗長データを追加する。L 層は 32Kbps の動画であるが冗長データを追加しないので、単純に 10mS に対して 1 パケットを生成し、1 パケットは 40B となる。

H 層データは 5 パケットのうちで 1 パケット紛失しても再生できるので、パケット紛失率 20%に相当する。また M 層は 20 パケットのうちで 2 パケット紛失しても再生できるので、紛失率 10%に相当する。

音声部分はデータレート 16Kbps に対してインターリーブブロック長を 100B と短くして遅延を 50mS に抑えているが、動画部分はデータレート 32Kbps に対してブロック長を 800B としているので遅延は 200mS となる。したがって、音声と動画は厳密には同期（リップシンク）しなくなる。

この設計の評価の１つとして、冗長データを含めた総転送量を計算しておく。冗長データを加える前のもとのデータ量は音声と動画を合計して 80Kbps であるが、10mS に 1 パケットの割合でパケット化 (パケット長 100B) してヘッダー (IP, UDP, RTP、合計 40B) を加えると 112Kbps に相当する。これはすべて L 層とみなして転送する場合に相当する。また、上で述べた傾斜優先度の場合は、総転送量は 123.2Kbps になる。ただし FEC 時はヘッダー長が 44B となる。すべてのデータに H 層と同じ 4+1 の冗長度を付加すると 135.2Kbps となる。

もう 1 つの評価として、実際のインターネット上でのパケット紛失パターンを本方式に適用した場合のパケット回復状況を、次の方法で測定中である。測定回線区間を定め、送信側から 10mS の間隔で上記のパケットに相当する長さのパケットを UDP にて送出する。受信側でパケットの到着を測定し、パケットの紛失パターンサンプルを取り出す。この測定をいくつかの特徴的な時間帯に行い、トレースデータとする。次に、このトレースデータをインターリーブブロックに分割し、ブロック内での紛失パケット個数を計数し、その数によって回復率を求める。現在測定作業中であり、データがまとまった時点で改めて結果を報告する。

第 4 章　鍵配送に基づくセキュアマルチキャスト

近年のインターネットの劇的な普及とインフラストラクチャの高帯域化に伴い，いままでは実現不可能だったアプリケーションが運用されつつある．そのなかでも特に 1 対多，多対多の通信形態であるマルチキャスト通信を利用するアプリケーションの開発が進められている．

これらを実際に運用していくためには，プライバシの保護や暗号化などのセキュリティの機構が必須となる．一般的にマルチキャスト通信においてセキュリティを提供するためには，マルチキャストグループに参加している全てのメンバが同一の暗号鍵を共有し，その鍵を用いてデータの暗号化，復号化を行なう．アプリケーションによってはグループメンバシップの変更ごとに共有鍵は更新されなければならない場合がある．暗号鍵の共有は手動でも可能であるが，規模拡張性のためにはメンバ間で自動的に鍵を共有するための鍵配送機構が必要となる．

鍵配送を実現する際には，トランスポートに対して以下の要素が要求される:

1. 配送の信頼性の提供

2. 遅延の少ない配送
3. メンバ間の鍵の一貫性の保証

しかしながら，既存のネットワークアーキテクチャではこれらの全てを満足することはできない．

そこで本稿では，要求 1，要求 2 を実現する機構として，**Reliable Multicast for Key Distribution (KDRM)** を，また，要求 3 を実現する機構として，**Key Consistency management Multicast Protocol (KCMP)** をそれぞれ提案する．KDRM では，誤り訂正符号に基づく Forward Error Correction (FEC) による損失回復と，ホップごとのタイムアウトによる再送方式を統合することにより，鍵配送のための遅延の少ない配送を実現する．FEC の冗長度と再送タイマのタイムアウト時間はその時のネットワークの状態に応じて最適な値を計算する．KCMP では，鍵とアプリケーションデータとの**因果順序**を考慮することにより，システムを停止せずに鍵の一貫性を保証する．

また本稿では，これらの機構の有効性を確認するための評価を行なう．KDRM については遅延と帯域のトレードオフをシミュレーションにより示す．また KCMP については基本性能を評価するためにプロトタイプを実装し，評価する．

そして最終的に，KDRM，KCMP を構成要素とするセキュアマルチキャストネットワークアーキテクチャを提案する．本アーキテクチャにより，セキュアなマルチキャスト通信が実現できる．

4.1 鍵配送の要求

セキュアなマルチキャスト通信の実現のためにはグループの全メンバで鍵の共有が必要となる．本章ではグループの共有鍵配送時にトランスポートに対して要求される要素について検討する．

4.1.1 配送の信頼性の提供

信頼性を必要とするマルチキャストアプリケーションにおいて利用される鍵の配送に必要な要素として，鍵配送の信頼性が挙げられる．グループのメンバ全てが鍵を共有できてはじめてアプリケーションデータのセキュアな配送が可能となるので，一部のメンバに鍵が配送されない状況は許されない．

信頼性のあるマルチキャスト通信を実現するために，現在までに多くのリライアブルマルチキャストプロトコルが提案されているが，鍵配送の信頼性はこれらのどのプロトコルによっても実現される [90, 91, 92, 97].

4.1.2 遅延の少ない配送の実現

AV 会議や分散対話型アプリケーション，データストリーミングなど遅延に敏感なアプリケーションでは，鍵配送の遅延はアプリケーションサービスの一時的な停止を意味する．例えばデータストリーミングにおいてあるホストへの鍵の配送が著しく遅延した場合，そのホストではストリームデータの復号化ができなくなる．したがって，リアルタイム性やインタラクティブ性を有するアプリケーションでは，遅延の少ない鍵配送が要求される．

現在提案されているリライアブルマルチキャストプロトコルは，アプリケーションに特化して開発されており，遅延の少ない鍵配送を実現できるプロトコルは少ない．またそのようなプロトコルが存在したとしても，効率の良い配送は望めない．したがって，遅延に敏感なアプリケーションのための鍵配送には，遅延の少ないリライアブルマルチキャストプロトコルが必要となる．

4.1.3 メンバ間の鍵の一貫性の保証

AV 会議やグループコラボレーション，分散対話型アプリケーションなどの送信者が複数存在するものでは，鍵の更新時のメンバ間での一貫性の保証が必要となる．メンバ間で鍵の一貫性が保証されていない場合には，一部のメンバでデータの復号化ができなくなり，完全なアプリケーションサービスを提供することが不可能となる．したがって，メンバ間の鍵の一貫性を保証することは重要である．

4.2 鍵配送のための RMT

本章では，前章で挙げた鍵配送の要求のうち，配送の信頼性の提供と遅延の少ない配送とを実現するリライアブルマルチキャストトランスポート (RMT) 機構である **Reliable Multicast for Key Distribution (KDRM)** の設計について述べる．また，シミュレーションによりプロトコルの評価を行なう．

4.2.1 KDRM の設計

KDRM の基本方針は以下のとおりである．

- 損失回復手法として FEC と ARQ を統合する
- FEC ではデータ長と損失率に応じて動的に冗長度を変化させる
- ARQ では ACK ベースで中間ノードが再送を行なう

KDRM は送信ノード，受信ノード，中間ノードの 3 種類のノードから構成される．送信ノードは鍵サーバが存在するノードであり，受信ノードはマルチキャストグループに参加しているメンバが存在するノードである．また，送信ノードから受信ノードまでのマルチキャスト配送経路上のノードが中間ノードである．

各ノードは，マルチキャスト配送木上での自分の上流ノードと下流ノードのアドレスを保持する．また下流ノードについては，Round Trip Time (RTT) とリンク損失率の平均値も保持する．上流ノード，下流ノードのアドレスは，管理者により静的に設定されるか，マルチキャスト経路制御表から取得される．平均 RTT は再送タイマのタイムアウト時間を，平均損失率は FEC の冗長度を見積もる際に利用される．

送信ノードの動作

送信ノードは以下のように動作する．

1. まず鍵サーバが鍵のデータを KDRM 層に渡す．
2. 下流ノードごとの FEC 符号化パラメータ n，k の値を計算し，鍵データを k 個に分割する．
3. FEC の符号化を行なう．FEC の符号器から生成された n 個のデータに KDRM ヘッダを付加し，データパケットを構成する．
4. データパケットのコピーを再送のために保持しておく．
5. 各下流ノードにデータパケットを送信する．このとき，下流ノードごとに RTT に基づく再送タイマをセットする．
6. 下流ノードから確認応答パケットが返送された場合には，対応する下流ノードの再送タイマを取り消す．さらに KDRM ヘッダのタイムスタンプフィールドから得られる値より RTT を再計算する．
7. 下流ノードからの確認応答パケットを受信しないうちに再送タイマが時間切れになった場合には，その下流ノード宛てに再送パケットを送信する．ただし，既に k 以上の確認応答を受信している場合には，再送は冗長となるので，対応するタイマをキャンセルし，データパケットを破棄する．なお，この場合は累積損失率の計算は行なわない．
8. 全ての下流ノードからの確認応答パケットを受信したら，再送のために保持していたデータパケットを破棄する．

中間ノードの動作

中間ノードは以下のように動作する．

1. データパケットを受信する．
2. 確認応答パケットを生成し，上流ノードに返送する．
3. 転送するデータパケットのコピーを再送のために保持しておく．
4. KDRM ヘッダのタイムスタンプフィールドを更新した後，各下流ノードにデータパケットを転送する．このとき，下流ノードごとに RTT に基づく再送タイマをセットする．
5. 送信ノードの 6〜8 と同様の動作をする．

受信ノードの動作

受信ノードは以下のように動作する．

1. データパケットを受信する．
2. 確認応答パケットを生成し，上流ノードに返送する．
3. FEC の復号化を行ない，フラグメントを再構成する．
4. アプリケーション層の鍵管理クライアントに鍵データを渡す．

4.2.2 シミュレーションによる KDRM の評価

KDRM により遅延の少ない配送が実現されることを評価するために，The VINT Project により配布されているネットワークシミュレータ (Network Simulator, ns)[94] 上に KDRM を実装し，FEC を用いた場合と用いない場合について評価を行なった．

シミュレーションはシミュレータ時間の 1 秒おきに 10 回鍵サーバが鍵を update したときの平均値を測定した．シミュレーショントポロジは図 4.1 に示した高さ 3 の n 進バランス木のトポロジで行ない，

n を 1 から 10 まで変化させた．n 進木の根のノードを送信ノード，葉のノードを受信ノードとし，その他のノードを中間ノードとした．また，全てのリンクは帯域が 100Mbps，遅延が 10msec として，シミュレーションを行なった．

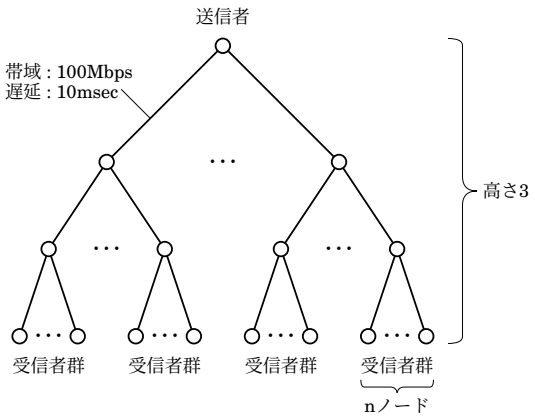

図 4.1　シミュレーショントポロジ

他のプロトコルとの比較を行なうために，シーケンス番号のギャップから損失を検出し送信者が再送を行なう PGM[97] の評価も行なった．PGM の ns への実装は Yunxi Shi による実装 [95] を利用し，FEC の有効性を確認するために本研究で提案する FEC の機構を拡張し，評価を行なった．

配送遅延

まずはじめに，KDRM，PGM における FEC による損失回復の有無での配送遅延の比較を行なう．本稿では，配送遅延を (4.1) 式のように定義する．

$$\text{配送遅延} = \frac{\text{実際の配送にかかった時間}}{\text{伝送遅延}} \qquad (4.1)$$

図 4.2，図 4.3 に，損失率が 2%の場合の KDRM と PGM でのグループサイズに対する配送遅延をそれぞれ示す．

それぞれのグラフでは，FEC による損失回復機構がある場合 (w/ FEC) と無い場合 (wo/ FEC) のプロットが図示されている．いずれのグラフにおいても，FEC を利用することにより配送遅延をほぼ 1 に抑えることができている．また，KDRM での配送遅延を PGM と比較した場合，PGM ではかなり大きな配送遅延の値になっていることが確認できる．これは，損失検出をシーケンス番号のギャップにより行なうことに起因する．

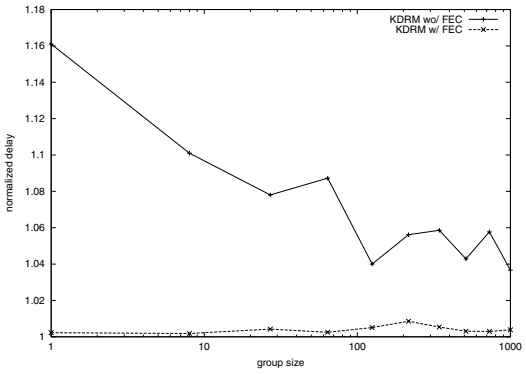

図 4.2　グループサイズに対する配送遅延 (KDRM)

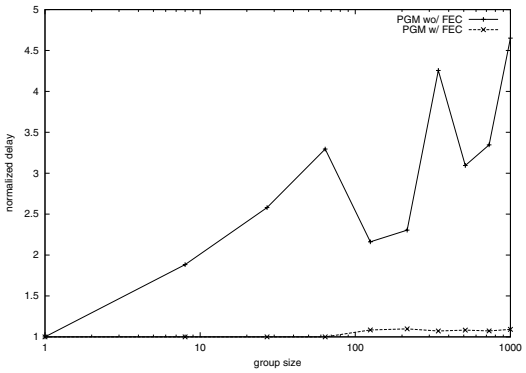

図 4.3　グループサイズに対する配送遅延 (PGM)

消費帯域

FEC を利用する場合，配送遅延と消費される帯域はトレードオフの関係にある．本節では，FEC を利用することによる消費帯域について評価する．

本稿では消費帯域を (4.2) 式のように定義する．

$$\text{消費帯域} = \frac{\text{全リンクで消費された帯域}}{\text{リンク数}} \qquad (4.2)$$

図 4.4 に，損失率が 2%のときの KDRM，PGM それぞれでのグループサイズに対する消費帯域を示す．

図では，FEC を利用する場合 (KDRM w/ FEC, PGM w/ FEC)，FEC を利用しない場合 (KDRM wo/ FEC，PGM wo/ FEC) ともに，消費帯域がグループサイズに依存せずほぼ一定値を取ることが確認できる．また，消費帯域は約 1.8 であることが確認できる．鍵のデータは，その鍵を用いて暗号化されるアプリケーションデータと比較すると，一般的に非常に小さい．したがって，FEC を利用する場合のオーバーヘッドは微小であるといえる．

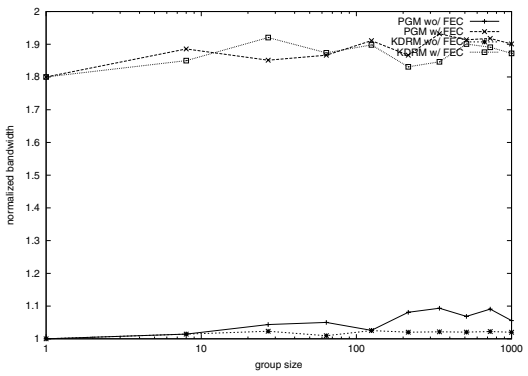

図 4.4 グループサイズに対する消費帯域 (損失率 2%)

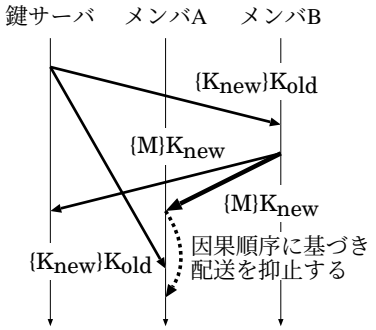

図 4.5 因果順序の例:メンバ A への鍵配送が遅延した場合

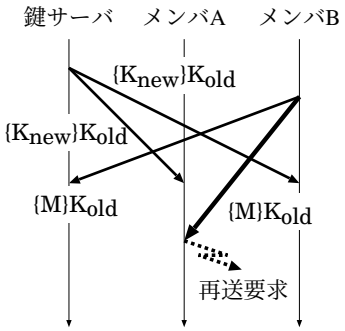

図 4.6 因果順序の例:因果順序が付加されない場合

4.3 鍵の一貫性を保証するトランスポート機構

本章では，メンバ間での鍵の一貫性を保証する機構である **Key Consistency management Multicast Protocol (KCMP)**[93] の設計と実装について述べる．

4.3.1 KCMP の設計

KCMP では鍵の一貫性を保証するために，鍵とアプリケーションデータとの**因果順序** [98] を利用する．図 4.5 と図 4.6 に因果順序を用いた配送の例を示す．

図 4.5 はメンバ A への鍵配送が遅延し，既に鍵を受け取ったメンバ B が新しい鍵 K_{new} で暗号化したアプリケーションデータ $\{M\}_{K_{new}}$ をマルチキャストした場合である．B から A に送信されたメッセージ $\{M\}_{K_{new}}$ に，そのメッセージが新しい鍵の受信後に送信されたものであるという因果順序情報が付加する．これにより A では $\{M\}_{K_{new}}$ を受信した際，新しい鍵 $\{K_{new}\}_{K_{old}}$ を受信しデータを復号化できるようになるまで，アプリケーションへのデータの配送を遅らせることができる．

図 4.6 は鍵とデータの因果順序が付けられない例である．この場合には，因果順序による順序付けができないので，メッセージの再送が必要となる．

KCMP の全体像を図 4.7 に示す．KCMP 自体は配送の信頼性は提供しないので，KCMP 層はリライアブルマルチキャスト機構の上位に，ともにトランスポート層のプロトコルとして配置する．そして KCMP の上位のアプリケーション層に，鍵サーバとマルチキャストアプリケーションが位置する．KCMP は，**ベクタタイムスタンプ機構**，**処理キュー**，**抑止キュー**の 3 つの要素から構成される．

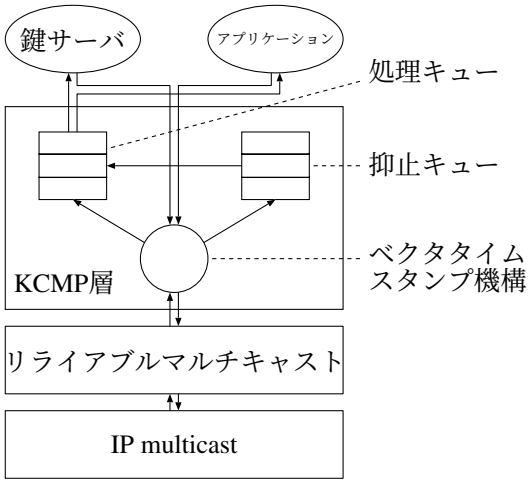

図 4.7 KCMP の全体像

マルチキャストデータを送信する場合には，鍵サーバ，マルチキャストアプリケーションともにベクタタイムスタンプ機構に送信データを配送する．ベクタタイムスタンプ機構では，図 4.8 に示す KCMP ヘッダをマルチキャストデータに付加し，リライアブルマルチキャスト機構に配送する．

マルチキャストデータを受信した場合には，データは全てベクタタイムスタンプ機構に配送される．

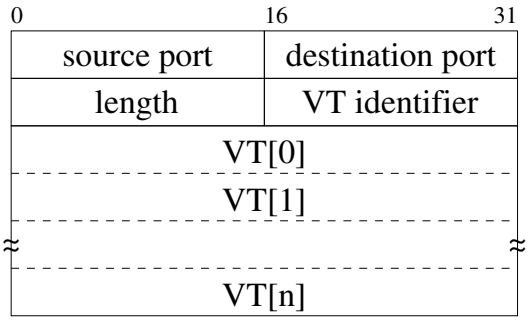

図 4.8　KCMP ヘッダ

ベクタタイムスタンプ機構では KCMP ヘッダ中のベクタタイムスタンプが検査し，そのデータよりも因果順序が前のデータを全て受信している場合には，そのデータを処理キューに配送する．そうでない場合には抑止キューに配送し，因果順序が満足されるまで待機させる．そして，因果順序が満足された時点で処理キューに配送する．処理キューでは FIFO 方式で鍵サーバあるいはアプリケーションにデータを配送する．

ベクタタイムスタンプ機構は，入力データ，出力データにより処理が異なる．以下ではプロセス p_i のベクタタイムスタンプを $VT(p_i)$ と表記し，プロセス p_i のベクトルの要素を $VT(p_i)[i]$ で表す．また，送信されるデータに挿入するベクタタイムスタンプを $VT(m)$ と表記する．

以下にベクタタイムスタンプの更新アルゴリズムを示す．ただし，n をメンバ数とする．

1. プロセス起動時に $VT(p_i)$ の全要素を 0 に初期化
2. 送信時に $VT(p_i)[i]$ を 1 増加
3. 送信データに $VT(p_i)$ を挿入 (これを $VT(m)$ とする)
4. p_i からのデータを p_j が受信した場合には (4.3) 式に従い $VT(p_j)$ を更新 (ただしメンバ数を n とすると $\{\forall k : k \in n\}$

$$VT(p_j)[k] = \max(VT(p_j)[k], VT(m)[k]) \quad (4.3)$$

ベクタタイムスタンプ機構では，マルチキャストデータの受信時に (4.4) 式の条件を満足していない場合には，マルチキャストデータを抑止キューに配送する．(4.4) 式の条件を満足している場合には，マルチキャストデータを処理キューに配送し，ベクタタイムスタンプを (4.3) 式に従い更新する．ただし，n をメンバ数とすると，$\{\forall k : k \in n\}$ である．

$$\begin{cases} VT(m)[k] = VT(p_j)[k] + 1 & k = i \\ VT(m)[k] \leq VT(p_j)[k] & k \neq i \end{cases} \quad (4.4)$$

なお，(4.5) 式の条件を満たす場合には，図 4.6 の状況が発生する．この場合には，送信者に対してデータの再送要求を行ない再送を待つか，アプリケーションに鍵の一貫性が保証されなかったことを告知する．ただし，n をメンバ数とすると，$\{\forall k : k \in n\}$ である．

$$VT(m)[k] < VT(p_j)[k] \quad k = i \quad (4.5)$$

4.3.2 KCMP の性能評価

KCMP の基本的な性能を評価するために，図 4.7 に示した KCMP のプロトタイプを FreeBSD 2.2.7-RELEASE 上に実装した．本節では KCMP の基本性能の評価について述べる．

パケット損失の評価

まずはじめに，通常のシステムと KCMP を統合したシステムでの，鍵の矛盾に起因するパケットの損失について評価を行なった．

評価は同一イーサネットに接続された 3 台のマシンにより行なわれた．3 台のマシンはそれぞれ鍵サーバ，データ送信者，受信者の役割を果たす．送信者はアプリケーションデータを 110 ミリ秒ごとに 100 パケット送信する．4.4 秒ごとに鍵の更新が発生し，鍵サーバが鍵を配布すると仮定する．

受信者において観測されたパケットシーケンスを図 4.9 と図 4.10 に示す．図の長方形は，受信ホストのアプリケーションにおいて復号化できた受信パケットを表す．横軸はアプリケーションがパケットを受信した時間である．なおこの例では，図 4.5 の状況が生じるように，受信者への鍵の配送を遅延させた．

図 4.9 は，通常のシステムでの受信パケットシーケンスである．鍵の更新が発生する 4.4 秒，8.8 秒付近で鍵の矛盾が生じている．鍵の配送が遅れているので，アプリケーションではこれらのパケットを復号化できず，破棄されている様子が観測できる．

一方，図 4.10 に示した KCMP を統合したシステムにおいては，抑止キューに溜められていたパケットが，鍵の矛盾が解消したときにアプリケーションに配送されているのがわかる．したがってアプリケー

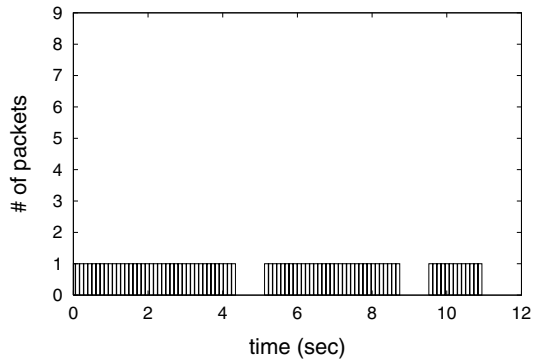

図 4.9 受信パケットシーケンス: 通常のシステム

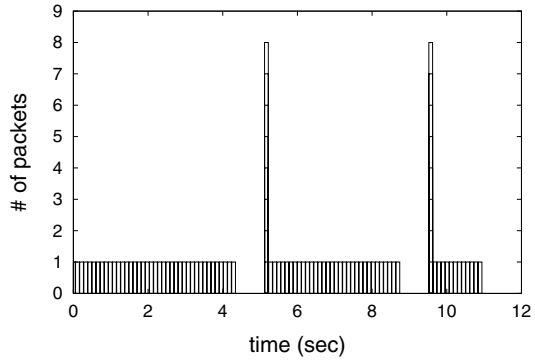

図 4.10 受信パケットシーケンス: KCMP を統合したシステム

表 4.1 UDP と KCMP での処理時間

	UDP (μsec)	KCMP (μsec)
input	16.7	17.0
output	9.2	13.6

ションは，全てのパケットを順番通りに受信することができ，またデータの復号化も可能である．

処理時間の評価

次に，KCMP 層と UDP 層での処理時間の評価を行なった．なお，KCMP では UDP の全ての処理をサポートしているわけではないので，UDP チェックサムの計算時間などを実際の処理時間から引いた値を UDP の処理時間とした．KCMP，UDP それぞれの input，output の処理時間を表 4.1 に示す．

input では両プロトコルの処理時間はほぼ同じである．KCMP におけるオーバーヘッドは，パケットに付加されたベクタタイムスタンプを比較するための時間である．

一方 output では，KCMP のオーバーヘッドはやや大きい．パケットの output 処理を行なう際には，ホストの保持する論理タイムスタンプを増加させ，ベクタタイムスタンプを KCMP ヘッダにコピーする必要がある．output 処理のオーバーヘッドはこのコピーにより引き起こされている．しかしながら，このオーバーヘッドはネットワークの遅延と比較すると無視できるほどの値であるといえる．

4.4 セキュアマルチキャストネットワークアーキテクチャの構築

本稿で提案するアーキテクチャを下に図 4.11 に示す．

マルチキャストアプリケーションのデータは，アプリケーション層でのセキュリティ機構，各アプリケーションに適したリライアブルマルチキャストプロトコルを通って，IP に渡される．共有鍵の一貫性が保証されなければならないアプリケーションの場合は KCMP も利用される．

鍵サーバが鍵を配送するときには，リライアブルマルチキャスト機構として KDRM が利用される．マルチキャストアプリケーションデータの配送と同様に，共有鍵の一貫性が保証されなければならない場合には KCMP も利用される．鍵サーバにより配送された鍵情報は，各セキュリティ機構に渡される．

本アーキテクチャにより、セキュアなマルチキャスト通信を実現できる．

4.5 まとめ

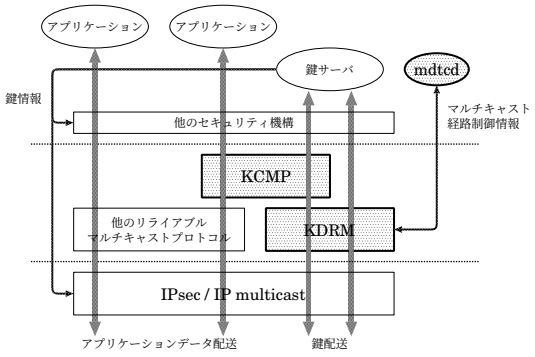

図 4.11 セキュアマルチキャストネットワークアーキテクチャ

本稿では，セキュアなマルチキャスト通信には既存のネットワークアーキテクチャでは不十分であることを指摘し，新しいアーキテクチャを提案した．

セキュアなマルチキャスト通信を実現するためには，グループのメンバ間で暗号鍵を共有する必要がある．鍵の共有は鍵サーバにより行なわれるが，この際の鍵配送のトランスポートに対する要求としては，以下のものが挙げられた:

1. 配送の信頼性の提供
2. 遅延の少ない配送の実現
3. メンバ間での鍵の一貫性の保証

既存の機構ではこれらの全てを満足することができないことを指摘した．

そこで本稿では，要求 1，要求 2 を実現する機構として，**Reliable Multicast for Key Distribution (KDRM)** を提案した．KDRM では，Forward Error Correction による損失回復と，ホップごとのタイムアウトによる再送方式を統合することにより，遅延の少ない配送が実現された．また，要求 3 を実現する機構として，**Key Consistency management Multicast Protocol (KCMP)** を提案した．KCMP では，鍵とアプリケーションデータとの**因果順序**を考慮することにより，システムを停止せずに鍵の一貫性が保証された．

また本稿では，これらの機構の有効性を確認するために評価を行なった．KDRM の遅延と帯域のトレードオフをシミュレーションにより示した．また KCMP の基本性能を評価するために実装して評価を行なった．

そして最終的に，KDRM，KCMP を構成要素とするセキュアマルチキャストネットワークアーキテクチャを構築した．本アーキテクチャにより，セキュアなマルチキャスト通信が実現できる．

第 5 章　DVTS over PGM 実験報告

本節では、否定応答ベースの高信頼マルチキャストプロトコルである PGM を用いて行なったデジタルビデオストリームのマルチキャスト実証実験について報告する。

5.1 はじめに

インターネット上のマルチキャスト通信に TCP のような信頼性を与える高信頼マルチキャストプロトコルは多数提案されている。

しかし、個々の高信頼マルチキャストプロトコルが提供する信頼性、実時間性といったプロトコル特性は網の損失率、トポロジー、ネットワーク構成に依存しており、これらの関連を解析的に解くことは困難である。

高信頼マルチキャストプロトコルの実用化を推進するためには、特定環境配下で動作する環境を構築し、詳細な問題を顕在化させ、同時に様々な環境要件に対するプロトコルの特性を解析し、プロトコル改定要求を抽出することが重要となる。

本実験では、動画像および音声のストリームデータの配送を行なうアプリケーションである DVTS を否定応答ベースの高信頼マルチキャストプロトコルで、受信者の数に対するスケーラビリティにすぐれる PGM[97] 上で動作する環境を構築した。本稿では、次節で提案されている高信頼マルチキャストの分類について次節で述べる。次に特定の損失状況で実稼働させるための、プロトコルパラメータの組合せについて紹介する。最後に、特定の損失条件の元で動作する適合型マルチキャストアプリケーションのアーキテクチャについて考察を行なう。

5.2 既存の高信頼マルチキャストの分類

高信頼マルチキャストプロトコルは多数提案されている。ここでは、損失回復の特徴に着目して次のように分類した。

- 肯定応答ベースで再送を行なうプロトコル
  送信者は、すべての受信者から肯定応答を受けとりその状態を管理する。パケットの損失は受信者から送信者に報告されて、送信者は再送を行なう。受信者の数に対するスケーラビリティを確保するためには、受信者を階層的に木構造に位置してフィードバックの爆発を防ぐ。中継者に受信者への配送責任を委任し ACK の集約を行なうプロトコルは、受信者の障害やネットワークの分割に対してうまく対応できる。肯定応答ベースのプロトコルは受信者の

数が増加するにつれて、送信者が管理すべき状態数や肯定応答を処理するための帯域などが増加するので最大スループットは劣化する。

- 否定応答ベースで再送を行なうプロトコル
  送信者は、受信者の状態を管理しない。受信者はパケットの損失を検知すると否定応答を発行してパケットの再送を促す。配送の信頼性を高めるために送信者がデータをメモリ (送信ウインドウ) から消去するまえに、ボーリングを行なうプロトコルもある。受信者の数に対するスケーラビリティは優れるが、受信者アプリケーションでは、配送が完全に行なわれない場合を想定した例外処理を行なう必要があるかもしれない。(アプリケーション要求による)

- 再送を行なわないプロトコル
  前方誤り訂正 (FEC) のように予め冗長パケットを送付することで信頼性を高めるプロトコル。肯定応答、否定応答を送信者へ返送する必要がないので、非対称経路のネットワークや再送によるトラフィック増加や再送パケットが到着するまでの時間が問題となる場合には有効なプロトコルと言える。

- ローカルリカバリに特徴を持つプロトコル
  IP マルチキャストの配送範囲を指定する機能[1]を用いて、再送要求の送付範囲を限定する事ができる。損失を検知した受信者はまず小さい配送範囲で再送要求をマルチキャストする。該当するパケットの受信に成功している受信者は再送要求に答えて該当するパケットを再送する。小さい範囲で返答する受信者がない場合は、配送する範囲を広げて再送要求を繰り返す。

5.3 既存の高信頼マルチキャストトランスポートの特徴

ここで既に提案されている代表的な高信頼マルチキャストの動作、特徴を見ていく。

[1] IPv4 では ttl 欄、IPv6 ではホップ制限欄、スコープ欄を活用する

5.3.1 RMTPII

Reliable Multicast Transport Protocol II (RMTPII)[101] は小数から多数への信頼性の良いデータ配送を行なう肯定応答ベースのマルチキャストプロトコルである。受信者を region という単位にグループ分けし階層化することにより木構造を形成する。各ノードにはノードを代表する制御ノードが割り当てられ、そのノードの受信者のメンバ管理、肯定応答処理、再送処理の責任を持つ。RMTPII はパケットの到着の信頼性を高く保証する。また、確実にパケットを受信した受信者の数を送信者に報告する機構を持つ。RMTPII では損失率や RTT(Round Trip Time) などの情報を用いて TCP の輻輳制御と互換性がある輻輳制御の機構を実現している。

5.3.2 PGM

PGM(Pragmatic General Multicast)[97] は、Cisco System 社が 1998 年の IRTF で発表した高信頼マルチキャストトランスポートプロトコルである。

PGM は否定応答を用いた再送制御により損失回復を行なうプロトコルで、到着順序が保証のされた、重複のない配送を行なう事ができる。PGM ではすべての受信者は送信元が送信するすべてのパケットを受信できるか受信できなかったことがわかることが保証されている。PGM はエンド・エンドで再送制御を行なうプロトコルで、ルータ支援を受けて、受信者の数に対するスケーラビリティを確保する。

PGM では表 5.1 に示す 5 つのパケット型が定義されている。

送信者は定期的に SPM を送出する。SPM を受けた PGM を解釈するネットワーク要素 (ルータ) は自らのアドレスを SPM に追加してマルチキャスト配送木の下流に SPM を転送する。受信者 (あるいは PGM を解釈する下流のルータ) は SPM に記載されているアドレスを参照して、自らの上流の PGM を解釈するネットワーク要素を認識する。送信者は、配送すべきデータを ODATA に載せてマルチキャストする。ODATA には通番が付与されている。受信者はこの通番の隔たりで損失を検知し自らの上流の PGM を解釈するネットワーク要素に対して NAK をユニキャストする。このとき付加的に TTL を 1 にして NAK をマルチキャストすることもできる。NAK を受けとった上流の PGM を解釈するネットワーク要素 (ルータもしくは送信者) は NCF を送出

表 5.1　PGM のパケット型

パケット型	意味
ODATA	Original Data。データを配送する
RDATA	Repair Data。再送データを配送する
SPM	Source Path Message。PGM 配送木を決定する。
NAK	Negative acknowledgement。否定応答。
NCF	Negative acknowledgement confirmation。否定応答受領通知

して NAK が別の受信者から送付されてくることを抑止する。NAK の送出は、バックオフしてから行なわれる。NAK 送出をスケジュールしている受信者が NCF もしくは同じ NAK を受けとると NAK の送出を一時取り止める。(バックオフする) 送信者が NAK を受けとると受信ウインドウに保持しているデータを RDATA に載せて再送する。

NAK,NCF,RDATA が損失した場合、受信者は NAK をバックオフしているのでタイムアウトを契機に NAK を再発行する。

RDATA が受信者に届けば、受信者はバックオフしている NAK の送出を取り止める。予め設定しておいた閾値の回数 NAK を再発行しても RDATA が得られない場合、受信者は回復不能エラーをアプリケーションに通知する。

PGM では、受信者はパケットを受信できるか、受信できなかったことを検知でするかいずれかの状態になることが保証されている。

PGM を解釈するルータは、NAK を受信すると修正状態を内部に保持して同じ NAK を 2 度以上受信しても上流の PGM を解釈するネットワーク要素へ送付しない。この内部状態は RDATA を受信したときに解除される。ルータは NAK を圧縮しトラフィックを削減するがパケットの再送に直接関与しているわけではないので多量のメモリを消費することはない。

5.3.3 FEC

前方誤り訂正 (FEC:Forward Error Correction) は、ネットワーク転送中に予期されるパケット損失する補完するための冗長パケットを送信し、受信側で、元のコードを再構成することで、高信頼なデータ転送を実現する技術である。FEC は、遠距離伝送で使われている誤り訂正符合化であるある Erasure Code を用いて実現する事ができる。

Erasure code について説明する。(n, k) ブロック Erasure Code は、k 個のソースパケット群から、その一部の k 個を取り出す元の k 個のソースパケットを復元できるような n 個の冗長パケット群を生成する符合化ブロックを示す。この符合化は、符合化したパケット群にもとのソースパケットが含まれる時に系統的 (systematic) という。パケットの損失率が低い時に系統的なコード化を行なったものの復号化の平均コストは系統的でない符合化を採用した場合と比較して低くなる。(もとのパケットの復号化計算コストは 0 である) また、符合化されるブロックが次のような行列を用いた線形変換で算出されるときに、線形 (linear) であるという。$\underline{x}$ を要素数 k の行ベクトル、 $\underline{y}$ を要素数 n の行ベクトル、G を $n * k$ の行列として、

$$\underline{y} = G\underline{x}$$

すなわち図 5.1 に示す式で表現され、G の任意の k 行の集合が完全な階数を持ち、任意の y の k 個の要素から x が再生できるとき線形であるという。

Erasure code は例えば字式で示される Vandermode Matrix を用いて生成できる。

$$g_{ij} = x_i^{j-1}$$

ただし、x_i は GF(p^r) [2]の要素。

FEC では、このような符合化を用いて作成された冗長パケットを送付するので損失が発生しても元のデータを受信側で再送無しに再生できる。(n, k) ブロックで符合化されたパケットから元のデータを再生するためには任意の k 個のパケットが正常に受信できていればよい。したがって次のような利点を持つ。

- 受信端末によって受信できたパケットにばらつきがある場合でも冗長の範囲内であればパ

[2] GF() はガロア体 (有限体) を表す

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} g_{11} & g_{12} & \cdots & g_{1k} \\ g_{21} & g_{22} & \cdots & g_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ g_{n1} & g_{n2} & \cdots & g_{nk} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_k \end{pmatrix}$$

図 5.1 Erasure Code の生成式

ケットの再送なしにデータを再現できる。

- 冗長の範囲を越えて損失が発生した場合、再送の仕組みを組み合わせて信頼性を向上させる方法が考えられる。このとき冗長化符合を再送すれば、受信側で受信できたパケットにばらつきがない場合でもばらつきがある場合でもデータを再現できるので効率的である。

欠点として次のようことがあげられる。

- バースト損失に弱い。すなわち (n, k) ブロックで符合化した場合、n-k 個以上の損失には対応できない。
- 計算時間がかかる。符合化復号化の演算時間は k を増大させると比例して大きくなる[3]。

5.3.4 SRM

SRM は、アプリケーションにとって意味のあるデータ単位で信頼性の確保をおこなうべきであるという ALF(Application Level Framing) の概念に基づき設計されている [92]。この考え方に基づき設計された SNAP(Scalabe Naming and Announcement Protocol) は、配送されるデータの単位に階層的な名前づけを行い、その状態を広告するプロトコルである。SRM は、この SNAP で定義される名前空間の状態を広告し、アプリケーションが名前空間に対応づけられたデータを共有するためのプロトコルである。SRM では、データの損失が検知されたときに、再送を行なう事で信頼性を確保する。SRM で信頼性を確保する単位は、ADU(Application Data Unit) と呼ばれている。ADU にはシーケンス番号が付与されている。損失はこのシーケンス番号の隔たりで検知する。論理的に階層的に配置された ADU を格納する器としてコンテナが定義されている。コンテナには CID が連続的に付与されており、この隔たりからコンテナの損失を検知することができる。しかし、この方法だけではコンテナ中の最後の ADU の損失を検出することができない。SRM では、各コンテナ毎の状態をセッションメッセージとして広告し、しっぽの損失を検知する。

損失検知がされたら、再送を要求する制御パケットを送付する。再送要求はランダムな時間が経過した後に repare request と呼ばれる制御パケットをマルチキャストする。待ち時間は、セッションメッセージのタイムスタンプにより求められたノード間の距離 $(d_{s,a})$ を元にして、次式で示される範囲の一様乱数として計算される。

$$\lfloor C_1 d_{s,a}, (C_1 + C_2) d_{s,a} \rfloor$$

ここで C_1, C_2 は定数である。このタイマが時間切れになる前に他のノードが先に時間切れを起こし同じ repare request がマルチキャストされた場合には、repare request は指数バックオフされ制御パケット送付が抑止される。要求されたデータを持つノードは、同様に距離にもとづいて計算されるランダムな時間待って repare packet を送付する。その範囲は上図の C_1, C_2 を D_1, D_2 で置き換えたもので表すことができる[4]。

ネットワークトポロジーと SRM の再送動作について説明する。SRM はパケット損失要求を行なったノードに一番近いノードが高い確率で応答する機構をもつ。SRM の特性を明らかにするために、上で定義した、C_1, C_2, D_1, D_2 およびネットワークトポロジの関係について解説する。

[3] [87] では、代表的な計算機システムの計算時間の最大値を紹介している。これによると Pentium 133MHz を搭載した計算機システムで FreeBSD 上の実装の復号化速度は、$\frac{9.573}{k}$MB/s である。

[4] ノード間の距離の測定には NTP で用いられている方法と同様の方法が用いられているので、ノード間で時刻が同期している必要はない。ただし、ノード間で転送されるパケットの経路が対称であることが必要である。

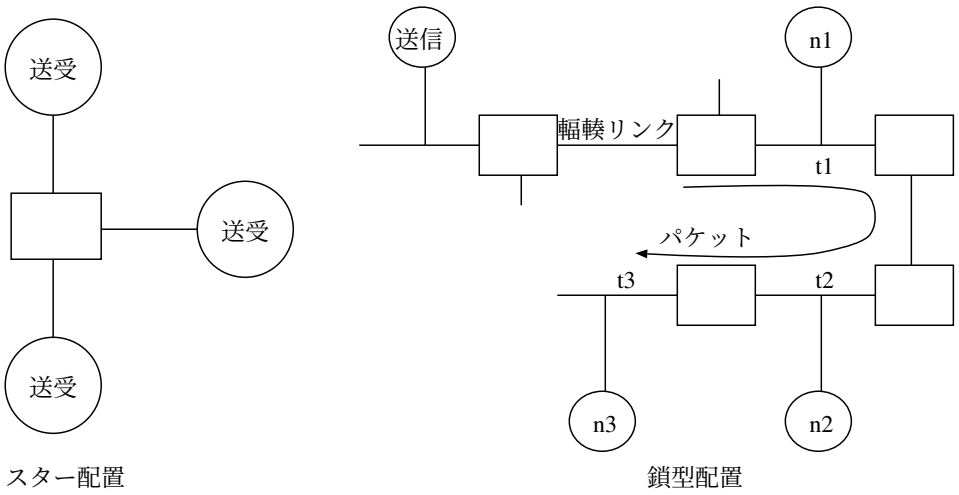

図 5.2　SRM のローカルリカバリとネットワークトポロジ

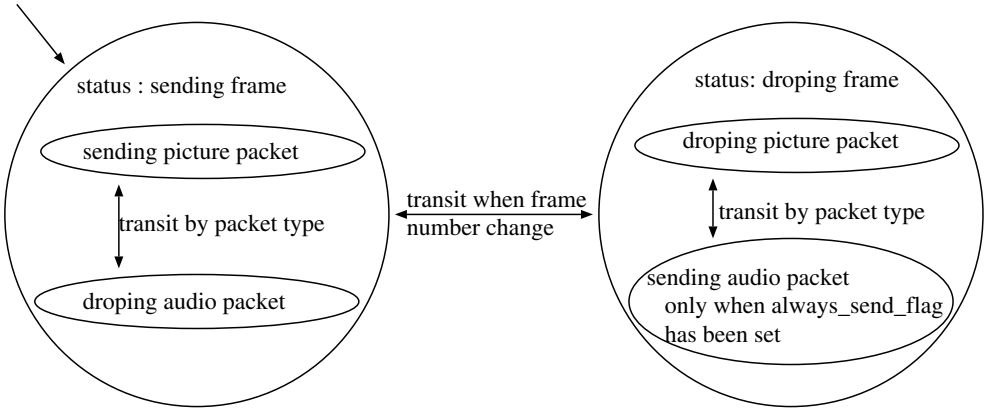

図 5.3　DVTS 送信者の状態遷移図

まず、ネットワークトポロジーとして均一な状態、すなわち、すべてのノードがスター状のトポロジーで接続されている状態では、すべてのノード間の距離が等い。したがって、同じ意味のパケットがネットワーク上に送出される機会を減らすためには C_2, D_2 を大きくすればよい。

次に、図 5.2 のようにノードが一列の鎖状に配置された例を考える。図では、あるリンクが輻輳しており、あるパケットが損失し、輻輳リンクを越えてシーケンシャル番号が飛んだデータパケットが到着している状態を表している。輻輳リンクに最も近いノード (n1) は、時刻 t1 に損失を検知する。次に接続されたノード (n2) はちょうどは、n1 と n2 の距離だけ遅れた時刻 t2 に損失を検知する。同様に次のノード (n3) は時刻 t3 に損失を検知するとする。C_1, C_2 がともに 0 の場合を想定すると、n1,n2,n3 がそれぞれ t1,t2,t3 に repare request を行ないネットワーク上に複数の repare request が送出される。C_2 を大きくする事で n2,n3 の repair request の重複を確率的に減らせる事がわかる。

最後にノード間の距離が不均一な例を考える。この場合 C_1(同様に D_1) を大きくする事でこれを抑止する事が期待できる。

SRM では、損失回復アルゴリズムの過去の振舞に応じてこれらの C_1, C_2, D_1, D_2 を動的に調整する適応アルゴリズムを提案している。

SRM は、広域ネットワークの不要なトラフィックを削減するローカルリカバリに特徴を持つプロトコルである。否定応答ベースのプロトコルなので損失回復に関して受信者が責任を負う。送信者はパケット

が到達したかどうか確認しない。SRM では、マルチキャストの配送範囲の制限機能を利用したローカルリカバリの機構を有する。損失を検知したノードは、まず local scope と呼ばれる範囲に repare request を発行する。バックオフしているタイムアウト以内に repare data が到着しない場合は、グループ全体に repare request をマルチキャストする。

SRM は、名前空間を共有するアプリケーションを広域に展開するのに適したプロトコルといえる。

5.4 DVTS について

DVTS は DV 形式の IEEE1394 フレームを IP データグラムにのせてインターネット上で配送するシステムである[5]。DV 形式は 6.3mm 幅の小型カセットテープを用いた大衆ビデオカメラのために設計された形式で、音声データ、画像データ、制御データを画面フレーム単位で管理する [89]。そのため画面フレームのコマ落しなどを容易に実現できる。DVTS は、画面単位のコマ落し機能や、画面フレーム、音声フレーム単位のバッファリング機構を有しネットワーク伝送帯域にあった配送を行なう事ができるシステムである。

5.4.1 フレームレート制御機構

DVTS の送信者は、画面フレーム、音声フレーム単位で送信を抑止する機構をそなえており、設定するフレームレート (何画面フレームごと伝送を行なうか) で配送を行なうことができる。また DVTS 送信者は、音声フレームの転送は抑止せず画面フレームの転送のみ抑止する機構を持っている[6]。送信者の状態遷移図を図 5.3 に示す。

設定するフレームレートと利用するネットワーク帯域の関係を表 5.2 に示す。数値は UDP データグラムとして送付するときの値である。

5.4.2 バッファリング機構

DVTS の受信者では伝搬遅延のジッタを吸収するためのフレーム単位でのバッファリング機構を導入している。映像フレームと音声フレームを別々に管理し、再生する時に合成することで、映像フレームの間引き転送が行なわれる場合でもフレーム単位で整合性のある画面の再生が可能となっている。この様子を図 5.4 に示す。

[5] http://www.sfc.wide.ad.jp/DVTS/
[6] http://www.sfc.wide.ad.jp/ akimichi/pv2000/

表 5.2 DVTS の消費帯域

設定フレームレート	IPv4 上での転送時 (Mbit/s)	IPv6 上での転送時 (Mbit/s)
1/1	30.47	31.70
1/2	15.72	16.83
1/3	11.48	11.84
1/4	9.01	9.33
1/5	7.54	7.83
1/10	4.74	4.87
1/20	3.26	3.39
1/30	2.79	2.90

表 5.3 ハードウエアの諸元 1

構成	仕様
CPU	AMD K6-III 400MHz
マザーボード	Freeway FW-T15VGF
チップセット	VIA Apollo VP3
FSB	100Mhz
主記憶	98,304KB
IDE HD	Quantum Fireball CX6.4A
NIC	DEC DE500-AA 21140A(de0)

表 5.4 ハードウエアの諸元 3

構成	仕様
CPU	Pentium II 266MHz
チップセット	Vendor=8086 device=7180
FSB	66MHz
主記憶	131,072KB
IDE HD	IBM-DHEA-36481
NIC	DEC DE500-AA 21140A(de0)
IEEE1394	Adaptec AIC-5800

5.5 実験環境

本実験では、動画像、音声のストリームの 1 対多への放送的な利用の実用性を検討する目的で、DVTS による動画像、音声のストリームの高信頼マルチキャストプロトコル PGM を利用した配信実験を行なっ

表 5.5　ソフトウエアの諸元 2

構成	仕様	所在地
OS	FreeBSD-3.2RELEASE	http://www.jp.freebsd.org/
PGM	kpgm-3.2R-990814.patch	http://www.iet.unipi.it/ luigi/pgm.html
DVTS	DVTS-0.2.0pgm.tar	lss5-is26:~emuramot/pgm-app/
IP マルチキャスト用アプリ	DVTS-0.2.1.tar.gz	http://www.sfc.wide.ad.jp/DVTS/

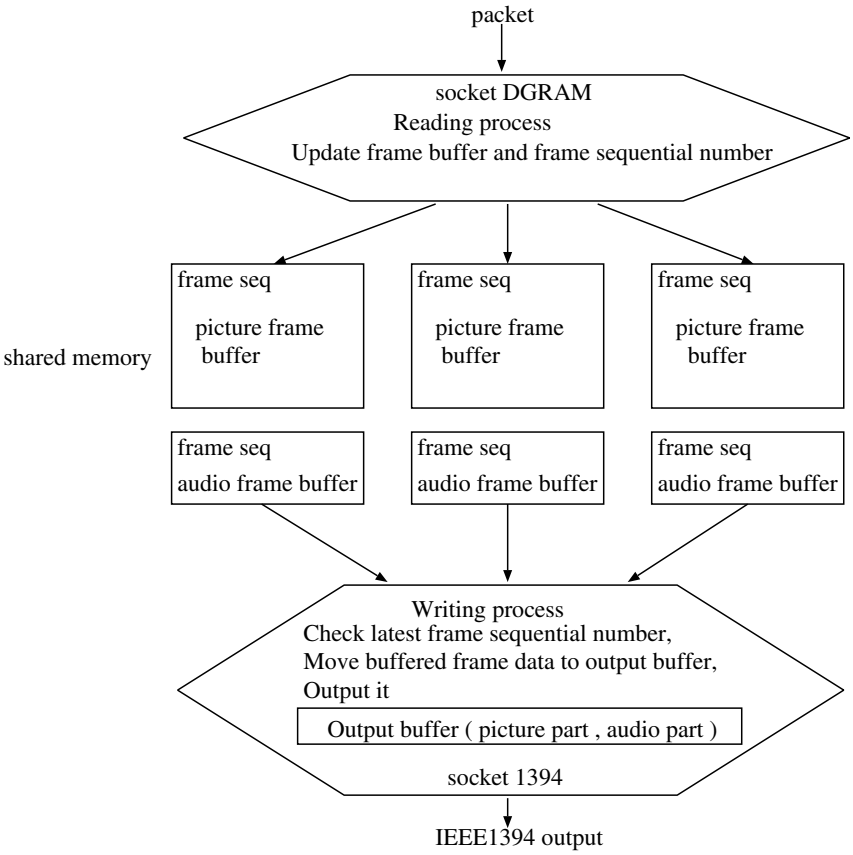

図 5.4　DVTS のバッファリング

表 5.7　DVTS 配送実験の結果

フレームレート	損失	IP マルチキャストによる配送	PGM を用いた配送
$\frac{1}{30}$	なし	画像、音声とも乱れ無し	画像、音声とも乱れ無し
$\frac{1}{10}$	なし	画像、音声とも乱れ無し	画像、音声とも乱れ無し
$\frac{1}{30}$	あり	画像、音声とも乱れ有り	画像、音声とも乱れ無し
$\frac{1}{10}$	あり	画像、音声とも乱れ有り	画像、音声とも乱れ無し

た。複数ストリームを合成するアプリケーションとして DVTS を採用した。また、受信者の状態を送信者が管理せず、1 対大多数の通信に向く否定応答の再送を行なうプロトコルとして PGM を採用した。実験環境を図 5.5 に示す。

5.5.1 損失を発生させる機構

本実験では、同一データリンクを用いて配送実験を行なった。実験系として必要な損失を発生させるために受信者に損失を発生させる機構を導入した。

具体的には、受信者のカーネルコード

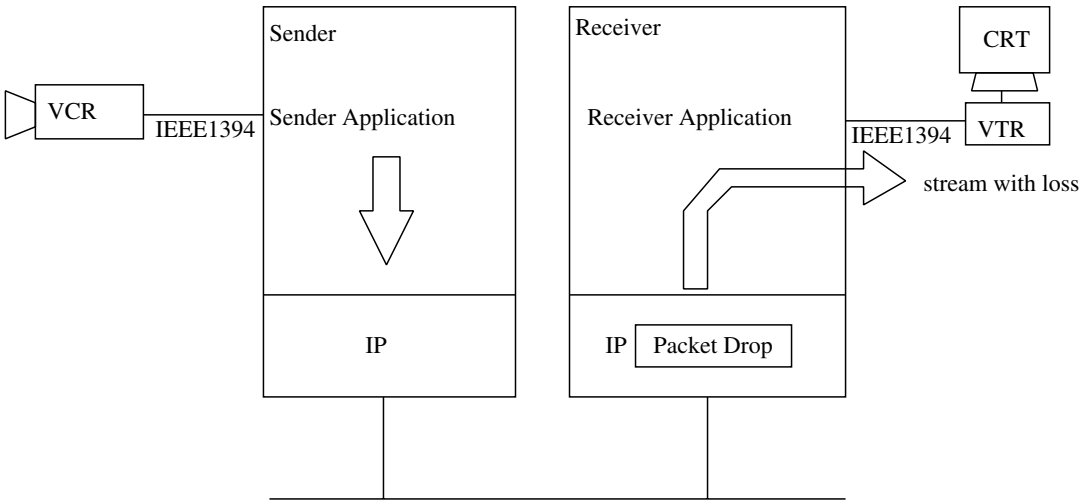

図 5.5 DVTS 配送実験環境 (PGM)

表 5.6 DVTS 配送実験時のパラメータ

パラメータ	送信者	受信者
pgm.sendspace	80000000	デフォルト値
pgm.recvspace	デフォルト値	5000000
pgm.nak_bo_ivl	デフォルト値	1
pgm.ncf_to_ivl	デフォルト値	デフォルト値
pgm.rdata_to_ivl	デフォルト値	デフォルト値
pgm.ncf_retries	デフォルト値	デフォルト値
pgm.rdata_retries	デフォルト値	デフォルト値
pgm.spm_ivl	2	デフォルト値
pgm.bandwidth	90000000	デフォルト値
pgm.odata_lifetime	1	デフォルト値
pgm.pgmcksum	デフォルト値	デフォルト値

の/sys/netinet/ip_input.c に乱数を用いて 500〜550 パケット受信ごとに 1〜3 つのパケットを落すコードを挿入した。

5.5.2 ハードウエア環境

本実験で使用した表 5.3 の諸元のハードウエアを送信者として利用した。受信者の諸元を表 5.4 に示す。

送信者の IEEE1394 ディバイスカードは、Texas Instrument 社の PCILynx を利用した。ビデオカメラは SONY 社の DVR-TRV5 を用いた。

5.5.3 ソフトウエア OS 環境

送信者、受信者とも利用した OS は FreeBSD-3.2R をベースに必要なカーネルパッチを適用したものである。使用したソフトウエアと入手元を表 5.5 に示す。カーネルパッチ適用後、/sys/netinet/pgm_usrreq.c を修正して、pgm.bandwidth の設定値の上限を 100000000 まで設定できるように修正した。

5.5.4 受信者、送信者設定

本実験で用いた設定値を表 5.6 に示す。

受信者では共有メモリを多量に用いた実験を行なったためカーネルコンパイル時に下記の設定を行ない取得できる共有メモリを量の上限値を増やした。

```
options "SHMMAX=(SHMMAXPGS*PAGE_SIZE+1)"
options  SHMMAXPGS=18000
options  SHMMIN=2
options  SHMMNI=33
options  SHMSEG=9
```

また送信者では多量の mbuf を消費するため以下の設定をカーネルに対して行なった[7]。

```
sysctl -w kern.ipc.maxsockbuf=95000000
```

5.5.5 実験結果

比較のため次の 2 つの実験を行なった。

- IP マルチキャストによる配送 (図 5.6 参照)

[7] 実際には、このパラメータを変更するために、カーネルコンパイル時の設定ファイルの maxuser を最大値 (512) にした。さらに、/sys/conf/param.c を修正した

- PGM を用いた配送 (図 5.5 参照)

について、それぞれパケット損失がない状況、パケット損失が 550〜650 パケット受信中、1〜3 パケット発生する状況で行なった。

フレームレート $\frac{1}{30}$ と $\frac{1}{10}$ での実行について、画質の状況、音声の途切れ、ノイズの発生状況を観測した。

結果、IP マルチキャストの配送では、損失がある場合画面上にブロック上の乱れが観測され、音声に雑音が観測された。PGM の配送では、画面は乱れずは観測されず、雑音のない音声が観測された。また、フレームレート $\frac{1}{30}$ の転送を損失のある状況で行ない 20 分以上の連続再生を行なえる事を観測した。(RDATA が 6 回連続で損失しなかった)

結果を表 5.7 にまとめる。

5.5.6 考察と今後の課題

DVTS は画像と音声、2 つのメディアのストリームを広帯域を利用して配信し、受信者でこれらのデータを同期させて再生するアプリケーションである。本実験を通じ、高信頼マルチキャストを用いてこれらを配送するに際し顕在化した問題点、課題をここで記述する。

ALF の必要性

本実験を通じて、適合型アプリケーションを容易に実現するため ALF[88] の枠組の必要性が顕在化した。ここでは、その理由を列挙する。

- デバッグポイントの確保
  私は、実時間アプリケーションの複雑性の回避のためフレーム境界の判定は送信者で行なうべきであると考える。実時間性プログラムのデバックや、時間的な因果率の崩れに起因するシステムバグの原因追求作業は、現象の再現が困難なため難しいとされている。具体的には、今回扱ったリアルタイムストリーム同期ずれのデバッグ作業があげられる。この作業はパケットの到着時刻と受信ウインドウメモリ状態、アプリケーションのプレイアウトバッファの状態を忠実に再現し解析を進める必要がある。フレーム境界の判定 (すなわちアプリケーションにとって意味のあるデータ単位 (ADU: Application Data Unit) の認識) を送信者で行なうと、この解析作業は送信者プログラムが完成した時点で行なうことができる。

- RTP 処理の容易性
  本実験のように受信者で画像、音声といった複数ストリームを合成同期させて画面のコマ落しのような処理を行なうためのプロトコルとして RTP が利用できる。DVTS において考えられる ADU の候補はつぎのようなものが上げられる。

  – 同期する画面フレーム+音声フレーム
  – 画面フレームと音声フレーム別々
  – IEEE1394 パケット単位

  損失率に応じてアプリケーションの提供する品質を制御する適合型アプリケーションでは、その要求に応じて適当な ADU を選択する。例えば音声優先といった要求を実現するためには ADU は画面フレームと音声フレームは別々の ADU として配送する必要がある。また、MPEG4 のコンテント符合化を用いて利用可能な帯域に応じて動画像の配送量を制御するようなアプリケーションでは、コンテントのデータフレーム毎に ADU を割り当てることが考えられる。

- DROP オプションタイムスタンプオプションの導入の容易性
  PGM では、配送するパケットの損失が検知されても再送要求しないことを通知する DROP オプションが提案されている。再送要求の必要のないパケットの次に送付するパケットに再送要求が必要な寸前のシーケンシャル番号を記載して送付する。受信者は、シーケンシャル番号の隔たりで損失を検知するが、Drop オプションが付与されている場合は、記載されている再送要求が必要な寸前のシーケンシャル番号より大きなシーケンシャル番号の欠落については否定応答を出さない。

  この場合、送信者アプリケーションがデータ送付時に否定応答の必要がない旨をトランスポート層に通知して送信命令を発行する。

  送信者でフレーム境界の判定を行ない音声フレーム、画像フレームの判定を行なっていれ

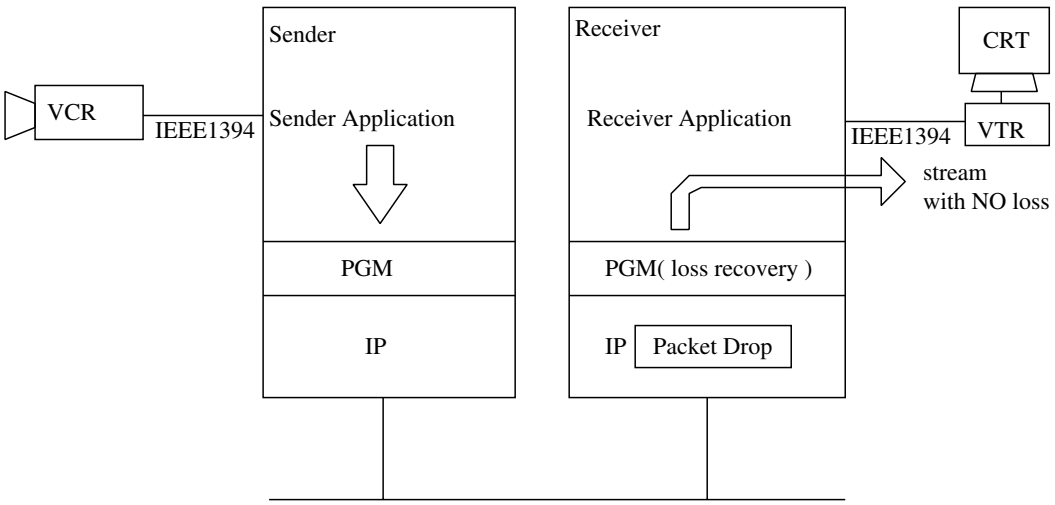

図 5.6 DVTS 配送実験環境 (IP マルチキャスト)

ば、送信時に否定応答の要否を指定することは容易である。また、PGM では、アプリケーションが必要とする時間までにパケットが到着する見込みがない場合、受信者が否定応答の送付を抑止するタイムスタンプオプションが定義されている。実時間処理を行なうアプリケーションでは RTP ヘッダにタイムスタンプを記載してデータを配送する。

これらのオプションをアプリケーションから容易に利用するためにも ALF の枠組に従った API を準備することが重要であると考える[8]。

高速化の技法と課題

DVTS は、フルレート転送を行なうと 35Mbps 程度の帯域を消費するアプリケーションである。損失回復を行なうために必要なメモリサイズは RTT が同じなら最大転送速度に比例して大きくなる。ここではアプリケーションの高速化で用いた技法について記述する。

- select システムコールの回避
  select システムコールは遅いことで知られている [99]。DVTS では共有メモリいて select システムコールを回避する方法を採用している。
- 高速化の課題
  本実験で用いた環境では、CPU 利用率が飽和するためフレームレート $\frac{1}{4}$ 以上の送信を 行なうことができなかった。カーネルの CPU 時間の利用率の分析 [9]の結果、mbuf の割り当てに関する操作が消費している時間が大きいことがわかった。本実験で用いた PGM 実装では、送信ウインドウを保持するためにカーネル中の mbuf を利用している。100 バイトを越えるデータを mbuf データを確保する時には、クラスタと呼ばれる外部バッファ領域が確保される。DVTS は高帯域を必要とするアプリケーションであるため、送信者の PGM では、多量のクラスタ必要とする。この割り当ておよび走査処理の高速化を行なえば、本実験で用いた環境で、より高帯域な配送を行なうことができる。

損失率と損失回復の方法

アプリケーション要求を満たすためには、単一プロトコルの強化拡張のみで対応することは最善とはいえない。損失率が小さく、バースト損失がないような場合は、3 つのパケットを 4 つのパケットに冗長符合化する IP-FEC[100] などで十分である。しかし、連続で 2 つ以上のパケットが損失するような状況では、PGM のような再送ベースの損失回復機構が必要となる。さらに損失が多くなる場合は、(輻輳などが原因と考えられるが) 階層符合化と RLC[96]

[8] Drop オプション、タイムスタンプオプションとも本実験で用いたプロトコルスタックでは、実装されていない。

[9] config -p でプロファイリングが可能なカーネルを構築することができる。

の枠組やその他の流量制御、輻輳制御の仕組みをアプリケーションに組み入れるべきかもしれない。また、極端に遅い端末などは、別セッションで再送するほうが望ましい場合もある。このように、マルチキャストアプリケーションが要求を実現するためには、その程度に応じてトランスポート層からアプリケーション層まで様々な技術を要求の程度に応じて使い分けることが重要となる。このイメージ図を図 5.7 に示す[10]。

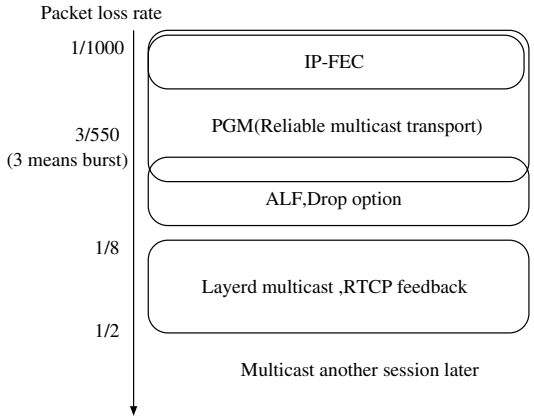

図 5.7　損失程度に応じた問題解決

統合サービスアーキテクチャが提供するサービスとの関連

統合サービスアーキテクチャで提唱されている制御負荷サービスでは、契約する帯域の範囲で低負荷状態のベストエフォット転送が保証される。これらのサービスを用いてその時点で利用可能な、制御負荷サービスのレベルに応じて画像転送の品質を切替える適合型マルチキャストのアーキテクチャの例を図 5.8 にしめす。

実時間性、信頼性が定義された socket を固定長のプレイアウトバッファを持ったアプリケーションが利用する様子をしめしている。

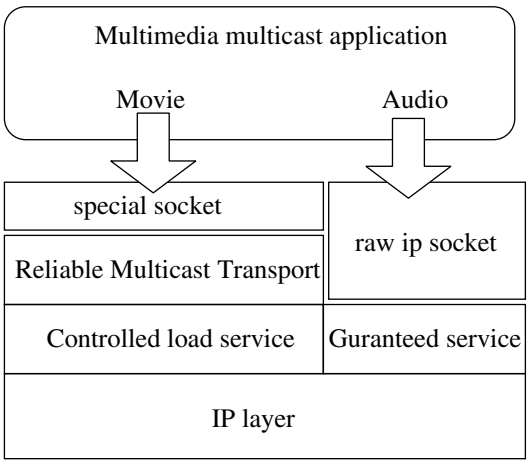

図 5.8　サービスクラスと適合型アプリケーション

[10] 図中の ALF 拡張とは、Drop オプション、タイムスタンプオプションなど含んだ拡張をさす。