第 10 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 の紹介について述べる．

1.1 背景

現在のインターネットにおけるマルチキャスト通信は，IP マルチキャストをベースに UDP を使っており，アプリケーションに対して信頼性のあるデータ通信は全く提供していない．そこで，広域環境に悪影響を与えず，ネットワーク資源を効率良く使用するリライアブル (信頼性のある) マルチキャスト機構をトランスポート層においてサポートする必要がある．リライアブルマルチキャストを必要とするアプリケーションは様々考えられるが，連続メディアをサポートするものとは異なり，主にデータの一貫性を必須とする．例としては，マルチキャストファイル転送，データベースやファイルシステムや WWW データのレプリケーション，WWW キャッシュ一貫性，対話型 virtual world の実現のためなどにリライアブルマルチキャストは利用される．

リライアブルマルチキャストに関する研究は以前から行なわれてきた．そのほとんどは，アプリケーション層での実現でメンバ数は比較的少数のものを対象としていた．それと比較して IP マルチキャストがサポートするメンバの数には限りがなく，すなわちスケールするように設計されている．トランスポートとしてリライアブルマルチキャストを実現するには多くの研究課題がまだ残っていると同時に，標準化できる機能もたくさんある．

1.2 IETF の動き

IETF では，リライアブルマルチキャストトランスポートの標準化を行なう動きが高まっている．2 年ぐらい前からは IRTF がリライアブルマルチキャストを取り上げていたが，プロバイダやベンダの影響力が多くなった今ではインターネットビジネスのためにはその機能は必須と認識されるようになった．

これに対し，IETF は 98 年 6 月に”IETF Criteria for Evaluating Reliable Multicast Transport and Application Protocols” という informational RFC を発行している．1998 年 12 月に行なわれた第 43 回目の IETF のプレナリセッションでは，特別に IRTF のリライアブ

ルマルチキャスト RG の紹介が行なわれ，その重要性が強調された．最近では 1999 年の 3 月の第 44 回目の IETF では，リライアブルマルチキャストトランスポートという名の BoF が開催された．ここでは IETF はどのようにリライアブルマルチキャストトランスポートをサポートして行くべきか検討された．WG を新たに設立するという結論はでなかったようだが，これは時間の問題であろう．

1.2.1 RMT BoF

ここでは，99 年 3 月に行なわれた RMT BoF について簡単に紹介する．

RMT BoF は基本的に IETF でリライアブルマルチキャストのためのプロトコルを開発するための WG を作成するためのディスカッションを行なうために開催された．リライアブルマルチキャストといってもさまざまな性質を考慮しなければいけない．そこで，IETF で取り上げるプロトコル/機能は以下のような前提を元にすることが明らかにされた．

- 送信者は 1 人または少数，受信者は多数
- 特に一つのプロトコルを作る必要はなく，いくつかのプロトコルを標準化することも可能

そして，具体的には以下の 2 点を目標として話合いは進められた．

- building block 方式と whole protocol 方式の可能性の検討
- RM Transport WG のチャーターを考えること

Building block (構成要素) 方式というのは，プロトコルのさまざまな機能単体を識別し，それを組み合わせることによって，特定なアプリケーションに適応したプロトコルを構築する方式である．このような方法をとって標準化を行なうとする場合，WG としては，種々の building block を識別し，それを一つ一つ標準化することになる．もう一方の方式の方が簡単で理解しやすいが，これは単にプロトコルを 1 から標準化することとなる．そこで，当日は，RMTP-II，MFTP，PGM などについて発表があった．これらについて本 WG のサーベイの章でも解説するのでここでは割合する．当日議論された building block すなわち要素技術として必要だと思われるものをここにリストアップしておく．

- Tree based ACK
- NACK
- E2e Nack suppression
- Router-based NACK suppression

- Data naming and framing
- Scalable session messages
- Packet-level FEC
- Congestion Control
- Security

これ以外に取り上げられた whole protocol 方式の例としては以下のものがあった.

- SRM: NACK based end to end
- RMTP-II: Ack-based with tree-based feedback aggregation non-aggregated ACK)
- PGM by CISCO: Nack-based with router support
- RMDP/RLC (Reliable Multicast Distribution Protocol): Layered with no feedback
- Starburst
- MFTP

最終的な結論，すなわち WG のチャーターについて，および WG の設立に関しては収束した話はなかった．かろうじてでた結論として，今後 building block 方式を中心に考えていくが，既存のプロトコルの開発を妨げないよう，すなわち開発速度を低下させないという条件で行なう．また，IETF ということで，リサーチは行なわないということが強調された．Building block の一つである congestion control はまだ研究課題の一つであって，IETF が積極取り込める状態にはなっていない．これについては年内までの IRTF の結果を元にし，その後 IETF で congestion control をどのように取り扱うか検討していくこととなった.

1.3　現状の問題点

リライアブルマルチキャストの標準化が難しい．それにはさまざまな問題点があるが，主にアプリケーションの要求が異なり，アプリケーション毎にリライアブルすなわち信頼性という用語の定義が違う．またこれに加え，既にあるインターネットのことを考慮しながら新しいプロトコルを導入する必要がある．信頼性を実現するためには何らかのパケット再送機構などが必要であるが，広域に分散した膨大のメンバ数をサポートするには，単純な ACK を用いる機構などは導入できない．ネットワークが ACK だけでパンクしてしまうからである．さらに TCP との親和性も重視しなくてはいけない．標準化の問題点は，RFC2357[81] に詳しくまとめてあるので参照して頂きたい.

1.4 WIDE RM WG

WIDE の RM WG の目的は，スケーラブルかつ異種性を考慮したリライアブルマルチキャストの実証的かつ実践的研究開発を行うことである．99 年の 1 月に発足され，99 年 3 月の合宿で初めて主要メンバ以外の参加者を含むミーティングを行なった．

現在のチャーターの中からその活動の概要を示す．DV クラスを使用した AV 通信，または大容量高速ファイル転送などを含む次世代アプリケーションからの要求を満足する新しいリライアブルマルチキャストプロトコルおよび基板技術を設計・実装・標準化を行う．また WIDE バックボーンを用いリライアブルマルチキャスト環境を構築・運営する．

また，WG の活動スケジュールは以下のようになっている．

- 1 年目
  - 既存技術を使った動く環境を構築
  - FEC などを使用
- 2 年目
  - 新しいリライアブルマルチキャスト技術を設計・実装
  - シミュレーションで評価，比較
- 3 年目
  - 高速ネットワーク (OC-3 以上) も含む運営
  - 標準化を目指す

1.5 報告書の内容

発足したばかりの WG なので，報告書執筆段階ではまだ成果になるものを提出できない．今回の RM WG のパートでは，既存のプロトコル SRM, MTP, RMTP-II のサーベイと今まで WIDE の中で行なわれたきた FEC に関する考察についてまとめた．最後には今後の実験計画も示している．

第 2 章

SRM (Scalable Reliable Multicast)

本章では，規模拡張性のあるリライアブルマルチキャストの一手法である **SRM (Scalable Reliable Multicast)** について解説する．

SRM ではパケット損失時に再送を行なうことにより，信頼性を提供している．パケット損失は，送信される各データにユニークにつけられたシーケンス番号のギャップから検出される．このシーケンス番号は，アプリケーションにとって意味のあるまとまりに分割されたデータ毎につけられる．これは **Application Level Framing** の概念に基づいたものである．しかしながら，シーケンス番号が最後のパケットを損失する可能性もあり，それを検出するために各メンバは，**セッションメッセージ**と呼ばれる全送信者のどのシーケンス番号までを受信しているかを示すメッセージを周期的にマルチキャストする．

以上の機構によりパケット損失を検出した場合には，再送を要求する制御パケットを送信する．無駄な制御パケットにより帯域を消費してしまうのを避けるために，SRM では **slotting and damping** と呼ばれる手法を採用している．各メンバは再送要求や再送を行なう際に，それらを引き起こしたメンバとの距離に比例したランダムな時間だけ送信を遅らせる．これらのパケットは，オリジナルのデータ同様グループ全体にマルチキャストされるので，複数のメンバが同じデータを損失していた際には，送信者から最も近いメンバのみが再送要求を行ない，他のメンバはこの再送要求を受信した際に再送要求を抑制する．オリジナルのデータのコピーを持つ全てのメンバは，この再送要求に答えることができるが，再送要求時と同様に，再送要求を行なったメンバに最も近いメンバのみが再送を行ない，他のメンバは再送をキャンセルする．このような手法を採用することにより，全てのメンバにおいて全てのデータを再送のために永遠に保持する必要がなくなる．

以下の各節では，SRM を構成する概念とその拡張について詳しく解説する．

2.1 セッションメッセージ

各メンバは，全送信者から受信したシーケンス番号の状態を報告するセッションメッセージを周期的にマルチキャストする．このセッションメッセージはセッションの現在の参加者を知るためにも使われる．セッションメッセージによる平均帯域消費量は，全体の一部分 (例えば 5%) に制限されている．しかし，セッションが大きくなってしまうと，扱い切

れないほどの状態が交換されてしまうので，SRM では状態空間を**ページ**と呼ばれる単位に分割し，現在参照しているページの状態のみを報告する．

状態の交換に加えて，セッションメッセージはノード間の距離を見積もるためにも用いられる．全てのセッションメッセージは，送信者の識別子とタイムスタンプを含み，任意の 2 ホスト間でセッションメッセージが交換された場合に，そのタイムスタンプからお互いの距離が計算される．この計算には同期した時計は必要ないものの，2 ホスト間の経路が対称であるという仮定が必要となる．

2.2 損失回復

各メンバにおいて損失が検出された際には，ランダムな時間待った後 **repair request** と呼ばれる制御パケットがマルチキャストされる．ここで用いられる **request timer** は，セッションメッセージにより見積もられたノード間の片道の時間の関数となり，$[C_1 d_{S,A}, (C_1+C_2) d_{S,A}]$ 間に一様分布する値として選ばれる．ここで，C_1，C_2 はパラメータ，$d_{S,A}$ はノード間の片道の時間である．このタイマが時間切れになる前に他のメンバのタイマが時間切れし，同じデータに対する repair request がマルチキャストされた場合には request timer は指数バックオフされる．

要求があったデータを持つメンバが repair request を受信した場合には，同様に $[D_1 d_{A,B}, (D_1+D_2) d_{A,B}]$ 間の一様分布である **repair timer** をセットし，時間切れになり次第 **repair packet** をマルチキャストする．ここで，D_1，D_2 はパラメータ，$d_{S,A}$ はノード間の片道の時間である．このタイマが時間切れになる前に他のメンバのタイマが時間切れし，repair packet がマルチキャストされた場合にはタイマをキャンセルする．

損失回復に対して受信者に責任を持たせることにより，再送データが正しく受信されたかを送信者が確認する必要がなくなる．

2.3 タイマの動的適応

SRM の損失回復の機構では，同じデータに対する repair request や repair packet が複数生じないように設計されているが，タイマの値に確率的な要素が含まれているので，これらの制御パケットの重複は起こり得る．request timer と repair timer のパラメータを操作することによりこれらの重複は減少させることができるが，これは再送されるまでの遅延時間とのトレードオフの関係にある．そこで SRM では，損失回復アルゴリズムの過去の振舞いに応じてパラメータの値を調節する適応アルゴリズムを提案している．

具体的には，repair request や repair packet を送信した際には C_1，D_1 をそれぞれ減少させ，再送の遅延時間を少なくさせる．また，これらの制御パケットの重複が検出された場合には C_2，D_2 をそれぞれ減少させることにより，重複する可能性を低くする．アプリケーションが望む値にこれらのパラメータを最適化することにより，動的な適応が可能と

なる．

2.4　ローカルリカバリ

上で述べた SRM の機構では，repair request も repair packet もグループ全体にマルチキャストされてしまう．したがって，パケット損失がグループのある一部分のみで起こっている場合には，帯域を無駄に消費してしまう．そこで SRM では**ローカルリカバリ**と呼ばれる手法を検討している．ローカルリカバリは，損失を共有しているメンバ数がグループの全メンバ数よりも断然小さい場合に行なわれる．ローカルリカバリでは，repair request がグループ全体ではなく，**local scope** と呼ばれる損失を共有している限定された範囲のみにマルチキャストされる．local scope 内でこのデータのコピーを保持しているメンバが存在せず損失回復ができなかった場合には，バックオフしたタイマが時間切れになった時点でグループ全体に repair request がマルチキャストされる．以上により，無駄な帯域消費を抑えることが可能となる．

2.5　まとめ

本章では，規模拡張性のあるリライアブルマルチキャストの一手法である SRM について概観した．SRM は現在，分散型ホワイトボードアプリケーションである **wb** をプロトタイプとして実装されている．wb のような，大規模でインタラクティブな要素を持つアプリケーションでは，SRM はその長所を生かすことができるリライアブルマルチキャストであるといえるだろう．

第 3 章

RMTP-II:Reliable Multicast Transport Protocol II

本章では,RMTP-II について解説する．一般的にひとつのリライアブルマルチキャストプロトコルでは, 様々なネットワーク形態の中で, 様々なアプリケーションに適切に対応することができないと言われている [102]．RMTP-II は,B.Whetten らによって開発された動画配送や新聞配信など少対多のアプリケーションに特化したリライアブルマルチキャストプロトコルである．以下では，RMTP-II を構成する概念について簡単に解説する．詳細は,[128] を参照.

RMTP-II(Reliable Multicast Transport Protocol II) は, 少数から多数へ信頼性のある効率の良いデータ配送を行なうマルチキャストプロトコルである．RMTP-II は, 対称性のあるネットワークや衛星とケーブルモデムなどで提供される非対称なネットワーク形式など, あらゆる形式のネットワーク上で動作する．また RMTP-II には, リライアブルマルチキャストトラフィックをモニタリングし、制御するネットワークマネージャが提供されている. RMTP-II の目的は, ネットワークマネージャに転送トラフィックを制御させながら, どのようなネットワークにおいても, エンドーエンドで遅延をできるだけ小さくし高い信頼性, 高いスループットを保証することである.

3.1 階層構造

RMTP-II は, 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 region には,region 内のノードを代表する特別なコントロールノードが割り振られ, そのノードが region 内の受信者のメンバ管理や受信者からの ack の受付,region 内でのパケットの再送などに責任を持つ．コントロールノードが,ack を受け取り集計し,region を代表して送信者へ再送要求をすることで, 同じ region 内からの再送要求の重複を避けている.

RMTP-II は, 受信者主導で信頼性を提供するマルチキャストプロトコルである．このプロトコルは, 現在の配送木を保持するために木を構成するメンバを完全に分散管理する．メンバを分散管理することにより送信者は, パケットを保持している期間や破棄するタイミン

グを簡単に決めることができる.RMTP-II は, パケットの到着という信頼性を高く保証し, 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. また, 付加的にパケットが落ちた時にのみ ack を返すネガティブアックをサポートすることもでき, 制御トラフィックを削減することが可能である.

3.2　輻輳制御

現在のインターネットは,TCP の輻輳制御への依存度が高い. 全てのストリームは,TCP の輻輳制御により帯域を公平に共有することができる. TCP と互換性のないトランスポートプロトコルが広く普及することは, インターネットに深刻なダメージを与える可能性がある. TCP と互換性を持たないリライアブルマルチキャストには, このことが障壁となりプロトコルの標準化が進んでいない. RMTP-II の輻輳制御は, 損失率や RTT(Round Trip Time) などを使い TCP の輻輳制御と互換性が取れるよう設計されている.

RMTP-II は, ネットワーク上を流れる全ての RMTP-II ストリームを集約的に管理することで, 輻輳制御をより完璧な物としている.RMTP では送信者は, 信用できる木構造の頂点のノードとやりとりをし, ネットワークの設定などを獲得する. 頂点のノードは,SNMP(Simple Network Management Protocol) インタフェースを持つネットワークマネージャを提供する. ネットワークマネージャは, 各ストリームを監視し使用帯域幅を決めることができ, また受信者が使用すべき輻輳制御パラメータを明示的に決めることができる.

これら, 自動的な輻輳制御と明示的な運用によりネットワークに悪影響を与えないリライアブルマルチキャストの配備が可能となる.

3.3　オプション

RMTP-II には, 以下の機能が補足的に提供されている.

- エラー訂正符号 FEC(Foward Error Correction)
- 受信者の送受信中の join/leave
- 全てのノードに SNMP のサポート
- 送信者, 受信者の締め出し
- リアルタイム性
- 動的な誤りの検知, 復旧
- ack による traffic 生成レートの制御

第 4 章

MTP: Multicast Transport Protocol

本章では，リライアブルマルチキャストの一つである **MTP(Multicast Transport Protocol)** について概観する．

MTP ではマルチキャストグループは **web** と呼ばれ，**master**，**producer** と **consumer** から構成される．master はメンバ間のメッセージの順序やレート制御について責任を持つ．レート制御は最低のフローを保てないメンバにグループからの離脱を要請することにより行なわれる．また，producer はデータを送信するメンバ，consumer はデータの受信のみを行なうメンバである．

producer はデータを送信する前に，master からトークンを獲得する必要がある．トークンを要求するパケットを producer から受信した場合には，master はシーケンス番号を含む確認パケットを返送する．各 producer がこのシーケンス番号をデータストリームに含めることにより，グループ内の全データストリームの順序を一意に決定することが可能となる．

通常トークンの発行は master に要求が到着した順になされるが，データに優先度を設けたい場合にはトークンの発行の順番を変更することができる．つまり，優先度が低いトラフィックに対するトークンの要求が先に到着していても，優先度が高いトラフィックに対するトークンの要求が到着した場合にはこのトークンが先に発行される．

MTP は NACK ベースのプロトコルである．consumer は，メッセージのシーケンス番号のギャップを検出した場合や，データストリームの受信が完了していないにもかかわらず **heartbeat** と呼ばれるある一定時間メッセージの到着がない場合には，producer に再送要求を行なう．producer は再送のために全てのデータのコピーを保持する．しかし保持しておくデータの量を制限するために，ある一定時間を過ぎたデータは消去される．

グループへの参加は，web に対して join パケットをマルチキャストすることにより行われる．web の各 master は，アプリケーションが指定するパラメータが許容可能なものであり，現在参加しているメンバのパラメータと互換性のあるものであれば参加を受け付ける．グループ参加時には，各メンバは master になろうと試みる．既に web に master が存在する場合には，そのメンバは producer か consumer として受け入れられる．しかし web に存在する master から返答がない場合には，そのメンバが新しい web の master となる．

MTP における一つの問題点として，全ての制御トラフィックが master を経由しなければならないという点が挙げられる．このような一点障害の問題を解決したバージョンとし

て，**MTP-2** [25] が提案されている．また他の新しいバージョンとして，**MTP/SO**[26] も提案されている．MTP/SO は MTP-2 を基本とし，グループを階層化する自己組織化機構が加えられており，ローカルで再送を処理することにより規模拡張性を提供している．

第 5 章

IP レベルの誤り制御プロトコル (IP-FEC)

5.1 概要

本章では IP パケットのロスを復元するための誤り訂正プロトコル (FEC : Forward Error Correction) について議論する。FEC は、広域環境における大規模なマルチキャストサービスを実現させるために、必要なモジュールとなることが認識されている。すなわち、FEC の適用により、より、大規模なマルチキャストサービスをより容易に実現することができる。

IP レベルで FEC による方法を用いるプロトコルであるので IP-FEC と呼ぶことにする。IP-FEC は、ネットワーク内で生じたパケット廃棄などの誤りを、再送による方法ではなく、FEC(Forward Error Correction) による方法で受信側が自発的に訂正するプロトコルである。我々は、IP-FEC プロトコルの仕様を検討し、BSD/OS3.0 のカーネル内に実装を行った。166Mhz のデスクトップ PC を 10Mbps のイーサネットで接続し、TCP/IP を用いて約 8kB のフレーム転送を行い送受信速度を測定した。誤りのない環境で 2.6 〜 5.0Mbps、誤り (パケット廃棄) のある環境で 1.9 〜 4.7Mbps のスループットが得られた。さらに、実ネットワーク上での実験として WIDE 研究会のネットワークと東芝間で性能評価を行った。パケット廃棄のパタンがパルス的な場合には、FEC がかなり有効であることが確認できたが、時間的に長くバースト的に廃棄した場合には、FEC 単独では不十分で他の方法、例えば ARQ や RED との組合せによりより効率的な誤り訂正が実現できるのではないかとの結論を得た。

5.2 はじめに

インターネットや M-bone の利用が増加し、end-to-end 間は地球的規模となり通信形態は、従来のポイント-ポイント型のみならず、マルチキャスト型が出現しようとしている。ネットワーク内で生じたパケット廃棄などの誤りに対しては、LAN 内で行なわれているのと同様に再送による方法で誤り訂正を行なっているのが現状であり、大規模なマルチキャストサービスの実現を行うには、再送オーバーヘッドの増加の問題が発生することが認識されている。

再送による誤り訂正は、リアルタイム性を要求されないメディアで端末間の距離が小さ

い場合、および受信者数が少ないマルチキャスト通信の場合には、有効であった。しかし、ネットワークが巨大化しデータトラヒックの変化予測が困難な場合、および受信者数が大きいマルチキャスト通信の場合には、FEC による誤り訂正がより適切であると考えられる。特に、FEC 技術は、次世代の誤りのないマルチキャスト通信には欠かせない技術である。

FEC による方法では、実データに対して予め誤り訂正コードを付加して送信し、転送中に誤りが生じた場合には受信側が訂正コードを用いて自発的に訂正する。よって、再送によるネットワークトラヒックの増加を防止することができ、送信サイドも再送要求に備えてデータを一定時間保存しておく必要がなくメモリの節約にもつながる。また、ARQ(Automatic Repeat Request) のみに基づいた再送方式は、遅延の増大、トラヒックの増加、コネクション管理の複雑化等から特に巨大ネットワークのマルチキャストには不向きである。しかし、FEC を用いるとより簡単に信頼性の高い通信を保証することが可能となるのである。

本章で提案する IP-FEC は、IP パケットのロスを復元するための誤り訂正符号を付与するプロトコルである。(パケット内のビット誤りを訂正する機能は付加していない。) なぜ IP パケットレベルであるかと言う理由は、それよりも下のレイヤに実装されていると転送途中にルータが存在した場合、そのルータが訂正プロトコルの終端/再生を行なわなければならなくなってしまい、負荷が急増してしまうからである。誤り訂正のアルゴリズム自体は演算処理がかなり多くハード実装されることが一般的であるが、本報告では BSD/OS 3.0 のカーネル内にソフト実装した。イーサネットで簡単な動画像を転送する程度のスループットが得られ、ソフトでも送受信ホストとして十分な速度を出せることが実証できた。しかし、ルータでの無駄なプロトコルの終端/再生は不必要なので、ルータに負担をかけず、end-to-end でデータリンクに依らないシームレスな誤り訂正を行なうにはレイヤ 3 のレベルで IP カプセル化した FEC を行なうべきであると考えた。

以上、述べたように基本的に IP-FEC はエンド-エンドで適用する方式であるが、ルータでも必要に応じて、終端もしくは符号化することも可能である。ルータで IP フォワーディングする際に FEC を適用し符号化し、受信端末の近くの FEC ルータで元のパケットに戻す、というトンネリングの手法を用いるものである。こうすることで、途中のルータ-ルータ間等で FEC を適用し、データの信頼性をより高めることができると考えた。

現在、IETF では、FEC をマルチキャストプロトコル内に組み込む、という方向性が打ち出された。実装したプロトコルを、実際のネットワークにより近い環境で運用試験を行なっていくことが急務である。そこでまず、WIDE 研究会のネットワークを用いてリアルトラヒック上での実験を行なった (ユニキャスト環境)。FEC の訂正能力には限界があり、それ以上のパケット廃棄には効果が全くなくなってしまうが、廃棄のパタンや連続した長さに依っては IP-FEC でも十分に対応できることが確認できた。しかし、ネットワーク上では様々な廃棄の状態が生じるため、FEC のみではデータの回復ができないケースももちろんある。そのような領域に関しては、FEC と ARQ、さらには RED ゲートウェイを組み合わせていくことが必要であると思われた。

以下、提案する IP-FEC プロトコルの仕様、実装の詳細 (プロトコルスタック、モジュール構成等) の説明、実験と性能評価を行っている。

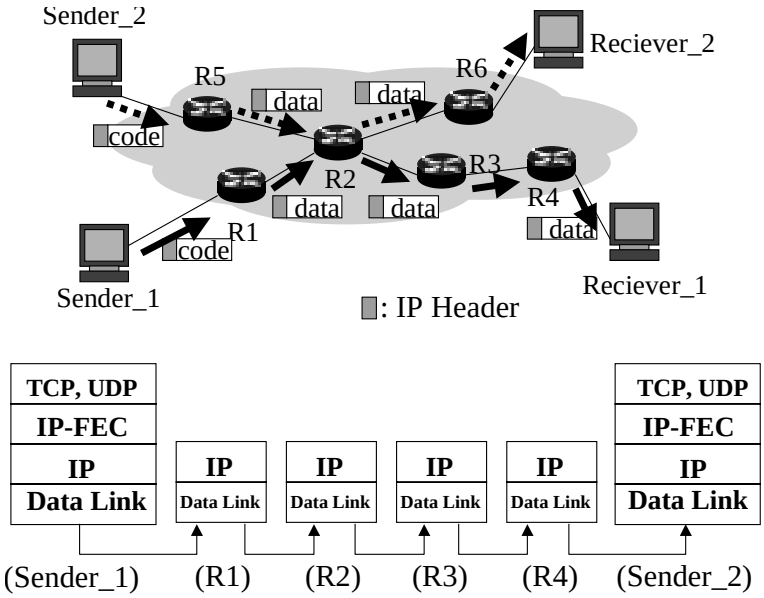

図 5.1: IP-FEC の使用形態

5.3 提案方式の仕様

5.3.1 使用形態の概要

IP-FEC はエンドエンド間の通信で用いることを基本としており、使用形態としては以下の図 5.1のようになる。

送信端末は実データの他に誤り訂正用のパケットを送信する。セグメント化された実データや誤り訂正パケットは IP データグラムにカプセル化されているので途中のルータでは一般の IP データグラムと同じようにフォワーディングされる。以下の節で送受信/符号化と復号化の手順について説明する。

5.3.2 送信の動作

図 5.2は送信の手順を示している。おおまかには、IP パケットをネットワークレイヤから IP-FEC が受け取り、誤り訂正符号を生成、FEC ヘッダの付与、さらに IP カプセル化し IP-FEC パケットとしてデータリンクレイヤに渡す、という動作を行なう。

1. original IP packet を interleave matrix に書き込み、誤り訂正符号である Reed-Solomon code を interleave matrix 内の code part へ生成する。

2. original IP packet に付与されている IP ヘッダ (IP-hdr) も含めて interleave matrix の縦方向に書き込んで行く。

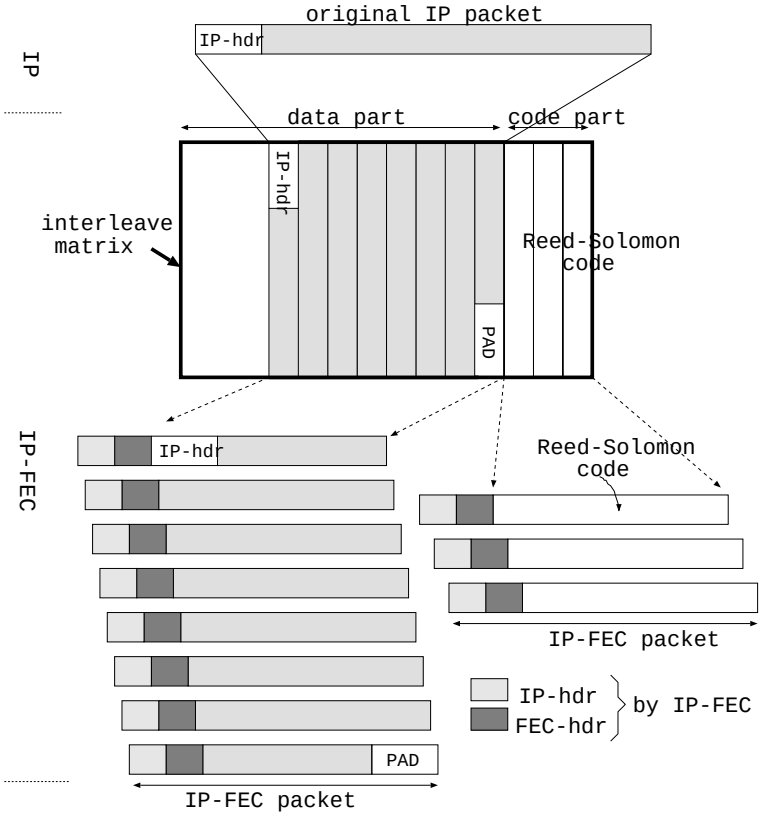

図 5.2: 送信処理

3. data part の最終列はパディング (PAD) を含む可能性がある。

4. 行毎に誤り訂正符号化の計算をし、code part の同じ行に書き込んで行く。interleave matrix の収量は可変であるが最大値が決まっている。縦のサイズ × 横のサイズ = 一定値で、ただし、横のサイズは最大で、255−code part 列数までである。Reed-Solomon 符号は、1 バイトを 1 シンボルとする誤り訂正符号。

5. data part の各列、code part の各列に対して、FEC ヘッダ (FEC-hdr) を付与する。FEC ヘッダは 4 バイトからなり、以下のフィールドを持つ (図 5.3参照)。(括弧内はビット数)

   - SN (8): マトリクス毎に初期化され付与されるシーケンスナンバ。同一のマトリクスに同じ SN を持ったパケットは存在しない。
   - #parity (4): マトリクス 1 行に対して何シンボルの訂正シンボルを付与しているかを示す。よって、各マトリクス毎に付与する訂正シンボル数が異なっていることも可能。

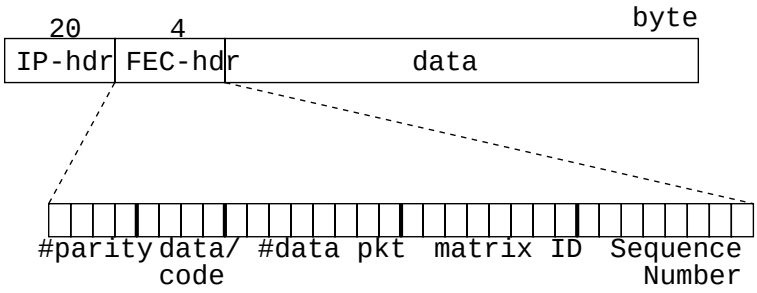

図 5.3: FEC ヘッダの構成

- data/code (4): そのヘッダが付与されているペイロードがデータを含む (0001) のか、それとも訂正コードを含む (0010) のかを示す。
- #data pkt (8): そのマトリクスが含んでいたデータ列数をしめす。
- Frame ID (8): マトリクス単位に送信側で順次インクリメントしていく ID。(IP レイヤはコネクションレス通信なのでパケットの到着順序が入れ替わることがある)

6. IP カプセル化後の IP ヘッダを付与する。基本的には上位レイヤから渡された時に付与されていた original の IP パケットのヘッダのコピーである。変更点は、プロトコルフィールドを FEC に設定、パケットのトータル長、チェックサムである。

7. 出来上がった IP-FEC パケットをデータリンクレイヤに順次渡す。

5.3.3 受信の動作

図 5.4に受信の手順を示している。data part、code part の各々から成る IP-FEC パケットは FEC ヘッダと IP ヘッダが付与されているので、一度 IP レイヤが受け取り IP-FEC へスイッチングすることになる。IP-FEC では、1 マトリクス分の IP-FEC パケットを処理し original IP パケットを組み立て、再度 IP レイヤに渡し、さらに上位のプロトコルへスイッチングしてもらう。このように受信側では IP 処理を 2 度通過することになる。

1. データリンクレイヤからネットワークレイヤに渡された IP-FEC パケットは、一般の IP パケットと同じようにスイッチングされ、ヘッダのプロトコルフィールドに FEC と明記されているので IP-FEC へ渡されてくる。

2. IP-FEC では、カプセル化の IP ヘッダを取り除き、FEC ヘッダをチェックする。matrix_id と seq_nm からペイロードの部分を適切な matrix の適切な列に書き込む。IP プロトコルの転送では正しい到着順序が保証されていないため、図 5.3で示されているように複数のフレームを同時に保持できることが必要となる。

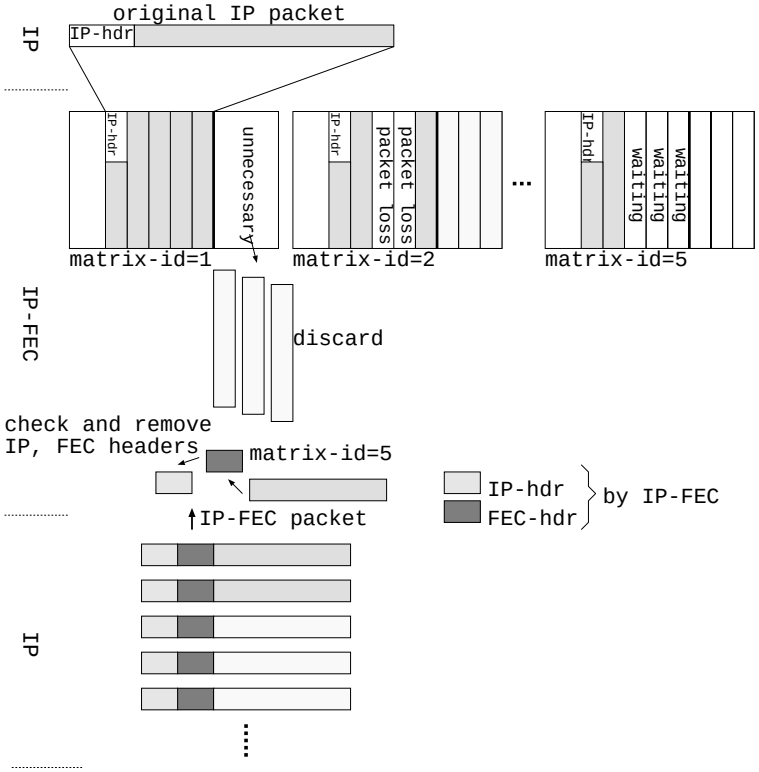

図 5.4: 受信処理

3. id=1 の matrix のように data part が全て揃えば original IP packet は組み立てが完了するので、code part の IP-FEC パケットである冗長符号は不必要なので廃棄する。

4. id=2 の matrix の場合は code part が到着しても data part が到着していないのでパケットロスした可能性がある。そこで誤り (消失) 訂正を行う機会を待つ。

5. パケットロスなしで original IP packet が出来上がった、もしくは、パケットロスはあったが訂正し original IP packet が再生できたら、再度、IP レイヤへ渡す。

6. パケットロスが多すぎて訂正能力を越えてしまっている場合は、訂正をあきらめフレームごと廃棄する。

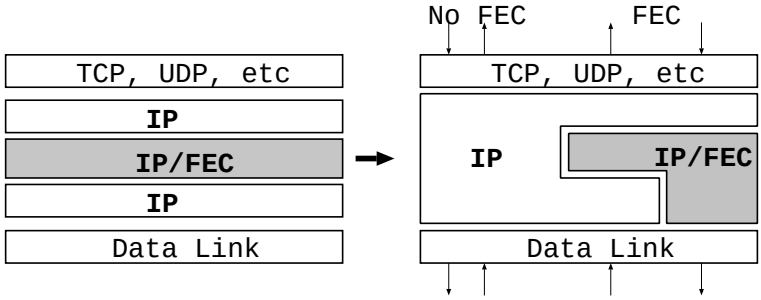

図 5.5: IP-FEC のプロトコルスタックと実装レイヤ

5.4 実装

5.4.1 概要

IP-FEC では、データ (IP データグラム) を複数パケットに分割しそれらを用いて誤り訂正パケットを生成する。送受信のレイヤ構造としては図 5.5のようになり、上位のトランスポートプロトコルにも非依存なので、多くのアプリケーションに適用可能なことも特長である。

以下、動作環境、ソースコードの動作について説明する。

5.4.2 動作環境

IP-FEC は BSD/OS Ver.3.0 のカーネル内に IP と隣接して実装されている。誤り訂正符号は RS 符号 ($G(x) = x^8 + x^4 + x^3 + x^2 + 1$) で、パケット廃棄のみを訂正する。現時点ではビット誤りは訂正しない。ハードウェアは、Pentium 166MHz の CPU を持つ、PC(PV3000 5166)。メインメモリ 16MB+ 増設メモリ 32MB を用いた。また、OS は BSD/OS Ver.3.0 である。

5.4.3 送信: ipfec_output.c

図 5.6に送信ホストのモジュール構成を示す。ipfec_output() は ip_output() からコールされ、完成した IP データグラムが渡される。(この IP データグラムをオリジナル IP データグラムとここでは呼ぶ) 受け取ったオリジナル IP データグラムを固定長にセグメント化し、FEC ヘッダ、続いて IP ヘッダを付与する。その際、IP ヘッダのプロトコルフィールドには FEC のプロトコル番号 (200) を書き込む。オリジナル IP データグラム (ヘッダも含む) 全体に対して誤り訂正符号を生成し、同様に FEC ヘッダ、IP ヘッダを付与する。こうして訂正符号を含む IP データグラムが生成されるが、その数は本実装では 1 〜 4 となっ

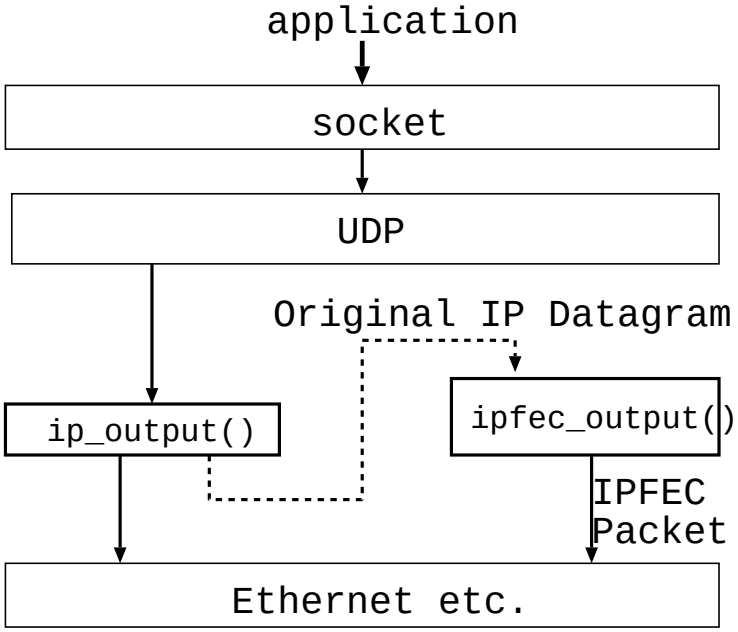

図 5.6: IP-FEC の送信動作原理

ている。

5.4.4　受信: ipfec_input.c

図 5.7に受信ホストのモジュール構成を示す。プロトコルフィールドに FEC と記してある IP データグラムは、IP のプロトコルスイッチで ipfec_input() へ渡される。ipfec_input() 内に持つマトリクスにパケットをためていき、データを含むパケットが全て到着したらオリジナル IP データグラムをリアセンブルする。それ以降に到着した訂正コードを含むパケットは捨てられる。

また、rx_fec_timer() がポーリングしており、タイムアウトの時刻になっても全てのデータパケットが到着していなければ、誤り訂正を行ないオリジナル IP データグラムを再生する。また、タイムアウト時間に達する前にデータグラムの再生が可能になった場合に、rx_fec_timer() は再生を開始する機能も持つ。ただし、パケットロスが多すぎて訂正不可能な場合には諦める。(到着した全パケット数 $dlen + clen$ がデータのパケット数 $dmlen$ 以上であれば、訂正可能な範囲である。)

リアセンブルし正しく再生されたオリジナル IP データグラムは、再度、ipintrq に渡され、正規の IP input の処理を施される。

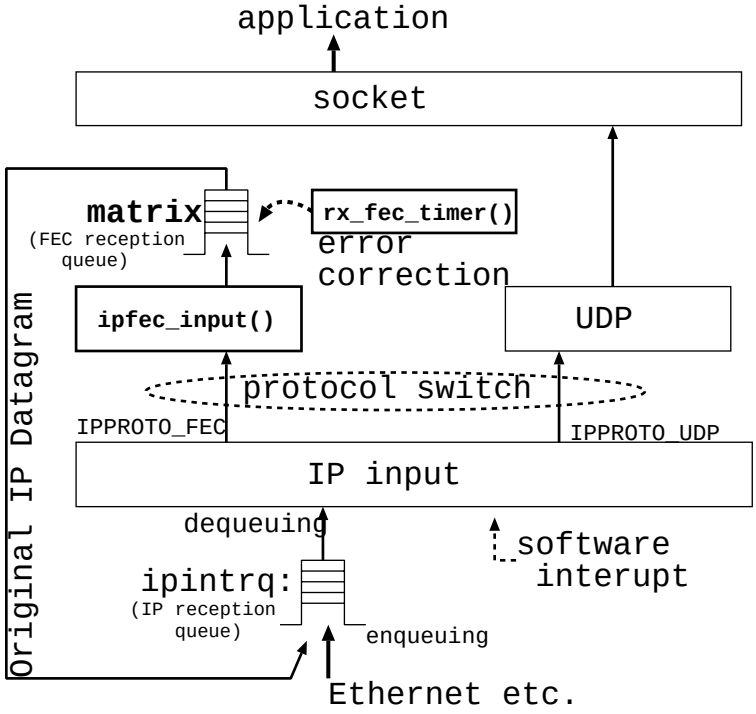

図 5.7: IP·FEC の受信動作原理

5.4.5 動作確認と基本評価

TCP/IP を用いて約 8kB のフレーム転送を行い通信速度を測定した。誤りのない環境で 2.6 〜 5.0Mbps、誤り (パケット廃棄) のある環境で 1.9 〜 4.7Mbps のスループットが得られた。(図 5.8参照)

誤りのない場合、要するにパケットロスを検出せずに data part の IP-FEC パケットを全て受信できた場合には、復号処理を行わない。いくつかの IP-FEC パケットをリアセンブルし、original IP packet に戻るだけであるので、処理速度は送信時の符号化の速度に依存することになる。誤りのある場合には、誤りの検出されたマトリクス単位に復号化の処理を行わなければならないので、受信側の復号化の処理速度に依存する。

より主観的な評価としては、音声通信や動画像転送にも適用して効果を確認し、リアルタイムメディアにも有用であることが分かった。

音声通信としては、vat(visual audio transport protocol) を用いて評価を行った。パケット廃棄がランダムに 5%発生する環境を想定し、FEC ありとなしを比較した。FEC なしではパケット廃棄によって生じる音声の途切れが目立ち聞き取りにくいが、FEC ありでは音声の途切れはおおむね回復し、人間の耳では廃棄のない状態と区別がつかないほど改善された。

動画像転送では H.263 を用いて、音声と同じようにパケット廃棄がランダムに 5%発生

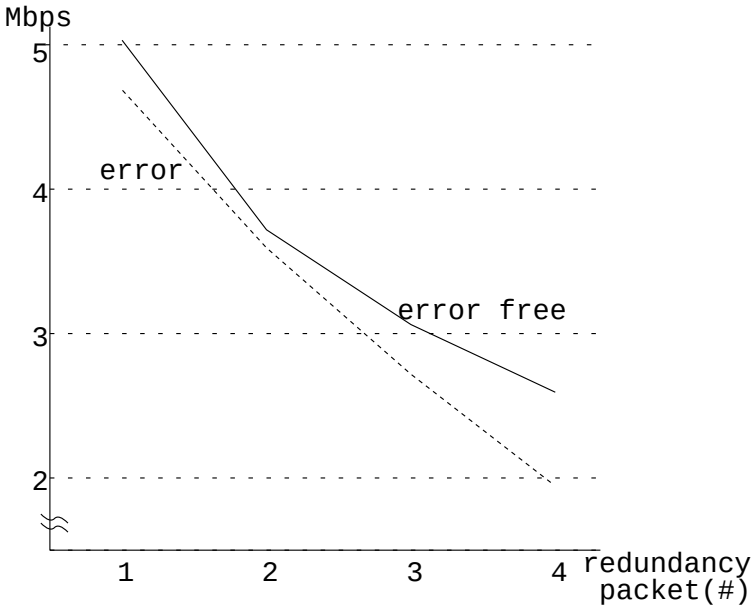

図 5.8: IP-FEC のスループット

する環境を想定し、FEC ありとなしを比較した。H.263 では、パケット廃棄が生じると画面にはブロック誤りとして表示され、しばらく連続すると画面が固まってしまう。がしかし、FEC を適用した転送ではパケット廃棄がない場合と同程度の画質で受信可能なことがわかった。

5.4.6　FEC ルータ

IP-FEC はルータ上でも動作するよう改良が加えられている。基本的に FEC はエンド-エンドで適用する誤り訂正方式であるが、それでは FEC を実装してない端末は全く FEC をすることができないということになってしまう。送受信端末間にはどのような品質のネットワークが存在しているかわからないが、ルータで FEC を行なうと、FEC を実装していない多くの端末も FEC を用いた通信を行なうことができ、その区間の通信品質を向上させることが可能となるのである。

FEC では、送受信端末が FEC の符号化/復号化ができることが必須条件であるので、FEC 送信マシン (FEC ルータ) が FEC の符号化を行ったら、必ず、FEC 受信マシン (FEC ルータ) で受信されなければならない。FEC の符号化を行うホストは FEC の復号化ができるマシンがカプセル化を解くよう、IP アドレスの宛先を変更しなければならない。

case (a) では、送受信双方のホストが IP-FEC を実装しており、一般的なケース。case (b) では、双方のホストの最寄りのルータが IP-FEC を実装しており、IP フォワードの際に符号化と復号化を行なう。case (c)(d) では、ホストの一方が IP-FEC を実装しているケー

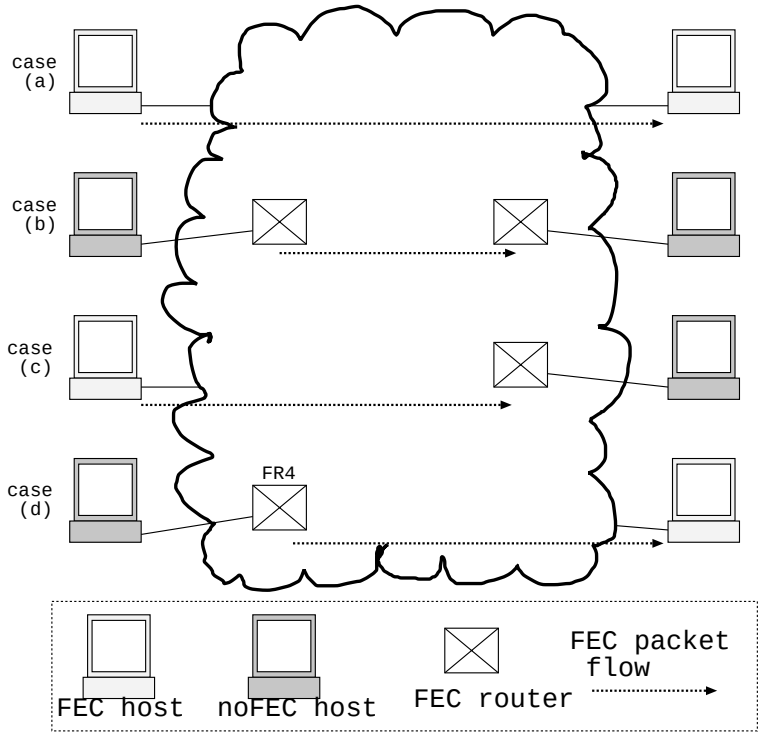

図 5.9: FEC ルータの使用形態

ス。次の節で示すが、このような機能を利用するためには予めコマンドでルータに設定しておく必要がある。

FEC ルータの作成により、これまで BSD/OS 上の限られたアプリケーションから、Windows マシンによる多くのアプリケーション実験（CTI、MPEG4 等）の可能性が開けることになる。

5.4.7 IP-FEC コマンド

IP-FEC はいくつかのパラメータが存在し、アプリケーションやネットワークの特性に応じて変更することが可能である。例えば、誤り訂正の強さ (= 重さ) やセグメントした場合のパケット長などである。

コマンドは ./fecset -[option] (引数) で、オプションは以下の通りになっている。符号化を行なう送信ホスト (ルータ) で使うコマンドと、復号化を行なう受信ホスト (ルータ) で使うコマンドとに分けられる。

送信ホスト (ルータ) のコマンド

-t のトリガは、6 つの引数並びが必要で、これはオリジナル IP パケットのヘッダと比較するものである。引数は、src-address, dst-addres, src-port, dst-port, ip-len, ip-proto。削

除も同様である。また、疑似アドレスとは FEC ルータとしてパケットをフォワードする場合に用いる。-p のトリガは、4 つの引数並び (A,B,C,D) が必要で、src-address=A かつ dst-addres=B ならば、src-address=C, dst-addres=D とカプセル化してフォワードするものである。削除もこれも同様である。

- -c FEC パケット数の設定 (1 〜 4)
- -l FEC パケット長の設定 (128 〜 1024)
- -t トリガの追加
- -r トリガの削除
- -g トリガの全削除 (引数なし)
- -p 疑似アドレスの追加
- -s 疑似アドレスの削除
- -d 疑似アドレスの全削除 (引数なし)

受信ホスト (ルータ) のコマンド

受信側ではセグメント化したパケットをストックしていって、到着が遅れているのか、廃棄なのかをタイムアウトで判断する。エンドエンドが時間のかかる場合は相応に大きな値を設定する必要がある。

- -o マトリクス毎のタイムアウト時間の設定

5.5　リアルトラヒック上での実験

WIDE 研究会では 3 月、9 月の毎年 2 回合宿が行なわれているが、その合宿ネットワークを利用したリアルトラヒック上で IP-FEC プロトコルの実験を行なった。

IP-FEC はパケット廃棄を訂正するエンドエンドのプロトコルであるが、研究所内のネットワークでは規模 (ルータ数、利用者数、トラヒック等) が小さいためかほとんどパケット廃棄は生じない。そこで、実装に伴う動作確認時には意図的にパケットを廃棄することで誤り訂正の動作を検証していた。このような背景から、より実際的で大規模な合宿ネットワークのリアルトラヒック上で実験を行なった。合宿ネットには合宿参加者 250 名程度がおり、各々が様々な実験/通信/生活を行なっている。

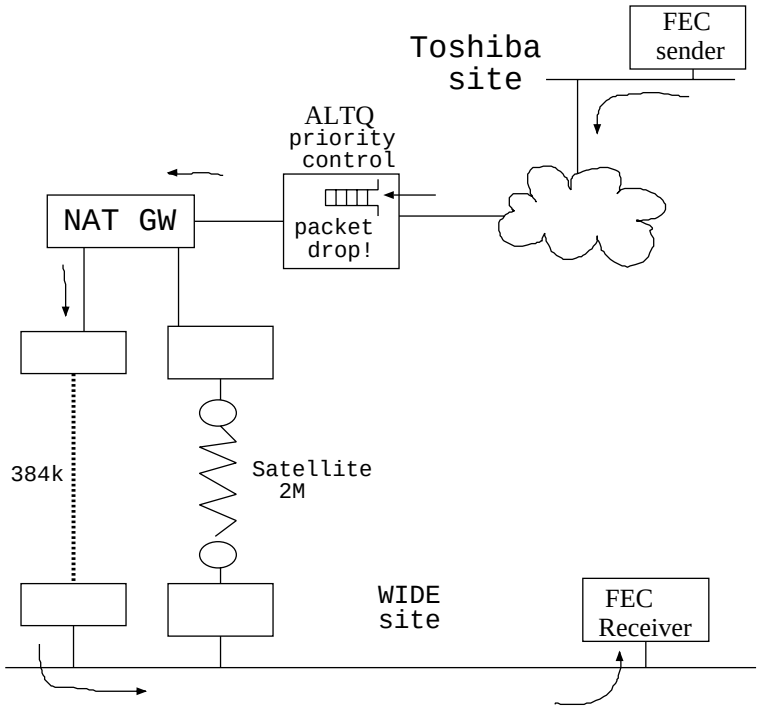

図 5.10: 東芝→ WIDE 合宿ネットワークのトポロジ

5.5.1 ネットワークのトポロジ

図 5.10に実験のネットワークを示した。実験では、東芝→ WIDE ネットへ UDP パケットを送信し、受信状態を観測した。東芝サイトと WIDE ネットに FEC を実装したマシンを 1 台づつ設置し、東芝サイトから FEC を適用した UDP パケットを送信し、WIDE ネット側で受信状態を観測するといったものである。通信経路は図 5.10の通りであり、パケットの流れを矢印で示している。

合宿ネットから外部へのゲートウェイでは ALTQ (Alternate queueing support for FreeBSD2.2. [1, 2, 5, 6]R) による優先制御が行なわれている。

5.5.2 評価方法

東芝サイトの FEC ホストから UDP パケットを一定の速度で送信し、合宿ネット内に設置した FEC ホストで受信状態を観測する。この東芝サイトというのは、社内のネットワークから出ている口で、128k の帯域を持っている。

送信ホストから 32k, または 16k の送信速度で UDP パケットを送信する。FEC では、1500 バイトのパケットを 3 つに分割し、訂正用の冗長パケットを数パケット (1,2,3,4) 生成し、続いて送信する。パケットの送信間隔はおおよそ 3msec ごとであるが、マトリクスの境界では 8msec かかっていた。マトリクスのタイムアウトは 1 〜 2 秒程度の長めの値に設

定した。NAT の付近には ALTQ というパケットの優先制御を行なうルータが介されており、基本的に FEC のパケットは最低の優先度で扱うようにした。要するに、キューが少しでも輻輳を始めたら FEC のパケットが真っ先に廃棄されるということである。

また、外部から合宿ネットに入って来るトラヒックは衛星を経由することになっているのだが、輻輳して自然にパケットが落ちる環境を作りたいので、地上線の 384k を利用した。(図 5.10参照)

5.5.3　測定結果

IP-FEC パケットが最低の優先度になっている制御下では、より優先度の高いコネクションが張られている間、長時間連続的に IP-FEC のほとんど全てのパケットが廃棄されてしまうことになり、FEC を適用している意味がなくなってしまうことが分かった。また、優先制御が行なわれていない場合 (全てのパケットが優劣なく同等に扱われる) についても受信状態を観測し、ネットワークが適度に混んだ時にはまずまずの効果があることが分かった。またバースト廃棄の前後では、50%程度の UDP パケットを訂正により復元させていることが確認できた。今回の実験ではスナップショット的な数値を測定した。各々のケースのパケット廃棄の状況を説明し、その際の FEC の誤り訂正成功率を示す。

- パルス的な廃棄 (廃棄の継続時間が短い) が数回生じたケース。

	エラーなし	訂正 OK	訂正 NG	成功率
1	81	5	0	100
2	1081	24	4	85

- バースト的な廃棄が生じたケース。具体的には TCP コネクションが約 10 秒程度はられたため。

	エラーなし	訂正 OK	訂正 NG	成功率
3	494	40	92	30
4	107	14	19	42
5	309	100	100	50
6	453	20	4	83

- 優先制御なし、3 時間分のトータル。16k で送信、冗長パケットは 2 個、タイムアウトは 3 秒に設定。

	エラーなし	訂正 OK	訂正 NG	成功率
7	13878	815	16	98

ネットワークのトラヒックは予想ほど混雑せず、定常的には廃棄が全く起きなかった。ただし、何らかの TCP コネクションがはられた時には大量データの移動が起こり、IP-FEC パケットは数秒から十数秒のバースト廃棄を余儀なくされた。また、http 等で生じると思われるパルス的なトラヒックによって、IP-FEC のパケットがパルス的に廃棄される。ALTQ で優先制御を行なっている場合、FEC パケットは最低の優先度に設定されているので、ほとんど廃棄されてしまうのである。

通信量が帯域以上になるような輻輳が継続する場合、UDP パケットが数十個まるまる落ちてしまうが、これらの誤りに関しては FEC による訂正は全く不可能である。よって、バースト廃棄が継続して起こっている状況で、IP-FEC が何らかの誤り訂正を行なえるのはバーストの前後の廃棄の部分であり、訂正を試みた場合の訂正成功率は概ね 50%である。小さなパルス的な廃棄 (= FEC の訂正能力内に収まる) の場合は訂正 NG がなく、完全に訂正できることも確認できた。

誤り訂正用の冗長パケットは特に明示のない限り 3 ないしは 4 パケット分生成し送信している。

5.5.4 実験結果に対する考察

今回の実験は制御と廃棄の特長から 3 つの状況に分類できる。

・優先制御ありでのパルス的な廃棄
・優先制御ありでのバースト的な廃棄
・優先制御なし

における、IP-FEC の振る舞いである。訂正確率からも分かるように、パルス的な廃棄にはかなり対応できるようである。パルスの高さが FEC の誤り訂正能力内ならば、もちろん完全に再生を行なうことが出来る。また、ここで数値として挙げていないが、まれに起こる小さなパルス的な廃棄には 100%回復できている。

二つめの優先制御ありでバースト的な廃棄の場合には、訂正成功率はかなり低いものとなってしまっている。これは、優先制御のために IP-FEC のパケットが大量に廃棄されてしまっており、訂正能力を完全に越えてしまっているものと思われる。数値として表示していないが、バースト廃棄では UDP パケットが数十個連続して廃棄されてしまっており純粋な FEC のみでは全く不十分である。

三つめは優先制御なしで、IP-FEC のパケットも他のパケットと同様に優劣なく扱われている。よって、廃棄量や、廃棄間隔は IP-FEC のレベルから見ると二つめの廃棄ほどバースト性は強くない。訂正成功率も 100%ではないがかなり高く、良好な状態である。廃棄の間隔が疎、よりランダムになるので FEC のアルゴリズムには好都合である。

7 つめのケースの測定値に注目してみる。ここには明示していないがアプリケーションデータ (同一マトリクスから生成され複数の IP-FEC パケットとなる) が全て廃棄されてしまったものが 135 であった。よって、7 のケースで FEC を行なっていない場合には、

$(815 + 16 + 135) \times 1500 = 144900$ バイトのデータが再送されていたはずである (1500 は実データサイズ)。しかし、FEC を行なっていたので再送は、(16+135)×(1500+1000) = 377500 バイトの再送となった (1000 はこの時の訂正符号サイズ)。FEC を行なっていると長期的なスタンスでは送信データ量は多くなることもあるが、時々生じるエラーのあった短期的な時間に注目すると再送量はかなり減少していることがわかる。

まとめると、FEC はランダムなパルス的な廃棄には強いので、バースト的な廃棄には他の方法との組合せが必要である。長時間のバースト廃棄では、アプリケーションレベルのパケット全体が数十個連続して落ちることもしばしばあり FEC のみでは手に追えない領域が出てきてしまうので、FEC と ARQ を組み合わせた制御が望ましい。また、バースト廃棄の継続時間をより短くランダムにするために例えば RED(Random Early Ditection) と共存させることなども有効であると考えた。

5.6　まとめ

IP パケットの廃棄 (消失誤り) を復元するための誤り訂正プロトコルについて方式提案、実装、ネットワークでの評価を行った。実効速度も簡単な動画像が転送できる程度のものが完成し、実ネットワーク上でも FEC がある程度有効な領域があることがわかった。実装している符号の性質に応じて、FEC の訂正能力には限界があるので、各々のネットワーク品質や、通信に要求される品質等を考慮し、適切なパラメータを選択していかなければならない。さらに、実験から得たことは、実ネットワーク上のトラヒックでは純粋な FEC のみで対応できないバースト廃棄もあるので、ARQ や RED などの他の技術と組み合わせていく必要があるということがわかった。

何らかのフロー制御とも協調できるように考えていかなければならない。

5.7　今後の課題

リアルタイム系のアプリケーションに適用した場合など、ランダムなパケットロスへの耐性がとても強いことはわかったが、実ネットワークでの実験では、ネットワークが輻輳してきた時、パケットが予想以上に連続してバースト的に落とされるので、FEC 単独では十分な誤り訂正ができないことがわかった。

ネットワークの環境としてある程度パケット廃棄がランダムに行われるような協調の組み合わせが必要と思われる。連続廃棄のバースト性をより小さく疎にするような機能が存在するとより効果が高くなるであろう。FEC を実装したホストの同じレイヤに FEC と連携する ARQ を備える、さらに、ネットワーク的には途中で実際に廃棄が起きるであろうルータで RED を機能させる、といった試みが当面の課題である。

また、訂正能力を高めつつ、ネットワークの輻輳を引き起こさない/ 激しくしないようなパラメータの調整は必須であり、FEC の効果をより高めるためにも重要な研究課題であ

る。FEC はデータに冗長性を持たせるためにパケット数が数倍に増えるが、データ量はそれほど増えていない。現状のルータの動作として、データ量の増加よりもパケット数の増加にシビアであるというところが FEC のメカニズムと整合性の悪いところである。実際の利用可能帯域を知り、それに応じたレートで符号化後のパケットを送信できるようなメカニズムに改善させることも考えていきたい。

第 6 章

むすび

本セクションでは RM WG の紹介，世の中の動向について解説し，現存するリライアブルマルチキャストプロトコルや機構についてサーベイを行なった．

6.1 今後の活動

RM WG の今後の活動については大まかに先に述べたが，ここでは詳しく WG の方向性，実験計画，研究計画について解説する．

リライアブルマルチキャストという研究テーマは，プロトコルの研究/提案/開発という流れを必要するが，その性質上，プロトコル階層モデルの下から上までさまざまな情報を統合しながら進めなければいけない．特に我々はアプリケーションを考慮する building block を構成する必要があることから，アプリケーションの要求などを深く検討する必要がある．そこで，他の WG との協調を重要視している．SOI WG を始めとした，データレプリケーションを必要とするアプリケーション，または www キャッシュなどのアプリケーション，またはセキュリティを重視するアプリケーションなどさまざまな要求を満たすフレームワークを提供することを目的としている．

その一方，高速リンクのサポートや，IPv6 におけるマルチキャストとの統合，単一方向の通信のみサポートする衛星も考慮する必要がある．すなわちアプリケーションの要求に加えて，下位層との統合も考慮する必要がある．現在，WG としては以下のような研究／実験項目を検討している．

- Building Block アーキテクチャの構築
- SOI や web cache に適した RM の実装
  - V6 対応の既存のプロトコルの WIDE 版を作成
  - これを使った運用／実験
  - FEC を導入したハイブリッドの作成
  - ハイブリッド版を使った運用／実験

  - 広域でのスケジューリング実験
  - 輻輳制御を考慮した見直し
  - 新たなプロトコルの作成

- セキュリティを重視する RM のプロトコル開発

これらの課題に取り掛かる前には，RM がサポートしなければいけないパロメータを明確にし，各々の適用範囲と対応するアプリケーションの要求を検討しなければいけない.