# 広帯域ストリーム伝送におけるソフトウェア損失回復制御の評価

近堂 徹[†] 西村浩二[††] 相原玲二[††]

インターネットの急速な拡大と広帯域化により，大容量のマルチメディアアプリケーションに対する要求が高まってきている．しかしながら，ベストエフォート型のネットワークでは QoS が保証されておらず，パケット損失が上位レイヤのアプリケーションの品質に影響を与えるため，FEC などのパケット損失回復制御が必要となる．CPU の高速化，PC の高性能化にともない，ソフトウェア処理による FEC が実現可能なレベルに達しつつあり，これにより，従来のハードウェアでは困難であった，より柔軟性を生かした制御が可能になる．本論文では，実ネットワーク上での伝送にパケット損失回復制御を適用する場合，ソフトウェア処理の観点からその効果と影響について明らかにする．損失回復性能評価と処理負荷および処理遅延の実測定から，効果的な適用方法について述べ，広帯域ストリームに対しても十分適用可能であることを示す．

# Evaluation of Software Loss Recovery Scheme for Broadband Real-time Streaming

Tohru Kondo,[†] Kouji Nishimura[††] and Reiji Aibara[††]

As the wide bandwidth networks are spreading, more and more Internet users are using multimedia applications. These multimedia applications, such as voice and video, have large volumes of data. However, since current best effort network can not guarantee QoS, packet loss has much negative effects on application quality. Although FEC is effective mechanism for multimedia transmission, we need to adjust parameters of FEC to improve loss resistance. Software FEC processing can work more flexibly as compared with hardware processing. In this paper, we describe application and evaluation of software FEC processing against packet loss over broadband Internet. We also indicate the effective measure from software processing point of view.

## 1. はじめに

ベストエフォート型のインターネットではエンドホスト間での通信品質が保証されておらず，IP ネットワーク上で発生するパケット損失や伝送遅延などが上位レイヤであるアプリケーションの QoS（Quality of Service：サービス品質）に影響を与える．QoS 制御はプロトコルの階層化と同様に複数のレベルとして定義することができ[1)]，ネットワークレベルでパケット損失率やジッタを限りなくゼロに近づけることで QoS を保証することも考えられる．しかし，これらの要求を完全に満たすネットワークを構築することは多大なコストが必要となり，インターネットの持つスケーラビリティと相反する．さらに，使用するアプリケーションによって要求条件も様々であることから，信頼性を必要とするアプリケーションで個別に適切な対策を行うことが重要になってくる．

一方，近年，ネットワークの高速化にともない，映像や音声といったリアルタイム性を有するマルチメディア通信が増加してきている．これらは，リアルタイム性実現のためトランスポートプロトコルとして UDP（User Datagram Protocol）を用いる場合が多く，データ伝送信頼性は保証されない．ゆえに，信頼性を確保するために，FEC（Forward Error Correction）[2),3)] や ARQ（Automatic Repeat Request）[4)]，これらを組み合わせることによる制御[5),6)] など，アプリケーションレベルでの QoS 制御が必要となる．

ARQ は，損失パケットを再送することによりデータを回復させる方法である．受信側で損失を検知すると送信側に再送要求をすることで回復することができるが，一方，受信者数が大規模の場合に応答が送信者に集中する可能性があり，スケーラビリティに乏しく，マルチキャストには適さない．FEC は，あらかじめ送信側で冗長なデータを付加して送信することで伝送の

† 広島大学大学院工学研究科
Graduate School of Engineering, Hiroshima University
†† 広島大学情報メディア教育研究センター
Information Media Center, Hiroshima University

途中で発生した誤りを受信側で損失回復する方式である．受信者ベースで回復処理を行うため空間的スケーラビリティに優れ，IP マルチキャストでのリアルタイム伝送に有効である．しかしながら，FEC の誤り訂正能力には限界があり，回復能力を超えるパケット損失に対応することができない．一方，必要以上に高い冗長度を適用することは処理負荷や伝送帯域の面から見て好ましくない．

PC の高性能化と CPU の高速化により，広帯域のマルチメディアストリーミングに対しても，従来ハードウェアで行われていた FEC がソフトウェア処理でも可能なレベルに達してきている．これにより，ソフトウェア処理の柔軟性を生かした制御が可能になり，パケットインタリーブの併用や FEC パラメータの動的変更が容易に実現できる[7),8)]．それにともない，これらの機能を用いた場合の効果と影響を明らかにするための新たな評価が必要となる．これまでにも，パケット損失を効率良く復元するための様々な手法が提案されているが，高ビットレートのストリーミングに適用した場合について十分な評価がなされていない[9),10)]．

また，TCP トラフィックが混在する環境下における MPEG 動画像伝送に対する FEC の適用方法についての研究が行われている[11)]．この研究ではインタリーブは考慮されていないが，TCP-Friendly なレート制御を取り入れたとき，FEC の冗長データの量を GOP 特性に応じて調整する手法が提案されている．これにより，TCP トラフィックが共存する現実環境下でも他のトラフィックと公平性を保ちながら，FEC 効果的に機能することが理論的に示されており，そのためにも柔軟な FEC 処理が必要とされる．

本論文では，FEC ソフトウェア処理の観点から，回復性能とその影響について議論する．実ネットワーク環境の特性から推定したロスパターンを用いて，バースト性を考慮したパケット損失に対する性能評価を行う．この結果は，ブロードバンドインターネット環境下でリアルタイムマルチメディアアプリケーションに FEC を適用する場合の 1 つの指標となると考えることができる．本論文の構成は以下のとおりである．2 章では，本論文で取り扱うパケット損失モデルと FEC について示す．3 章では，実環境におけるパケット損失のトレースを採取し，ロスパターン特性を決定する．4 章では，決定したロスパターンを利用して実験を行い，FEC の適用方法と回復性能について示す．最後に 5 章でまとめを行う．

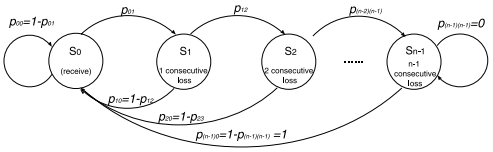


**図 1** Extended Gilbert モデル
Fig. 1 The Extended Gilbert Model.

## 2. パケット損失モデルと FEC

### 2.1 パケット損失モデル

ネットワーク上のパケット損失の特性に関する研究は，これまで様々な報告がなされている[12),13)]．パケット交換ネットワークにおけるパケットの到着確率は，公衆電話網などでみられるポアソン過程ではなく，Self-similar（自己相似性）で Long-range dependence（長期依存性）の特徴を持つことが，これまでの解析結果として示されている．ここで，自己相似性とは，パケット損失や到着間隔などのバースト性が観測するタイムスケールによらないことを意味する．このような環境下では各パケット損失に時間的依存関係があり，バースト性を持つパケット損失が発生すると考えられる．FEC は，パケット損失の分布がその回復性能に大きく影響するため，評価を行うにあたりこの点を考慮する必要がある．

RFC3357[14)] に，インターネット上のパケット損失に関する特性を見極めるために，連続してパケット損失する長さを表す “loss period”（**損失長**）とパケット損失の発生間隔を表す “loss distance”（**損失間距離**）の 2 つの基準が定義されている．なお，本論文では，連続して損失するパケット損失をバースト損失と定義する．

この点を考慮したモデルの研究が，これまでなされている[15)]．本論文も，この 2 つの要素を取り扱ったロスパターンを考えることで，ブロードバンド環境におけるバースト性を持つパケット損失をシミュレートする．

バースト損失のモデルでは，Gilbert モデルや Markov モデルなどが用いられているが，本論文ではバースト損失を適切にモデル化できる Extended-Gilbert モデルを採用する．Extended Gilbert モデル[16)] は**図 1** に示すように，過去の $n$ 連続で損失した状態により現在の状態が決まる，つまり，$P[X_i|X_{i-1}toX_{i-l}\text{all lost}]$ となるモデルである．

この推移行列は以下のようになる．

$$P = \begin{bmatrix} p_{00} & p_{10} & \cdots & p_{(n-1)0} \\ p_{01} & 0 & \cdots & 0 \\ 0 & p_{12} & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & p_{(n-1)(n-1)} \end{bmatrix}$$

よって各状態確率 $(\pi_0, \pi_1, \ldots, \pi_{(n-1)})$ は次のように計算される．

$$P \times \begin{bmatrix} \pi_0 \\ \pi_1 \\ \vdots \\ \pi_{(n-1)} \end{bmatrix} = \begin{bmatrix} \pi_0 \\ \pi_1 \\ \vdots \\ \pi_{(n-1)} \end{bmatrix}, \quad \sum_{i=0}^{n-1} \pi_i = 1$$

Extended Gilbert モデルは損失長のモデルであり，損失間距離に関しては考慮されていない．ある FEC のパラメータを決定したとき，その回復性能は発生するパケット損失の間隔に大きく依存するため，同じ平均パケット損失率でも損失間距離の違いにより回復性能も異なってくる．よってこの点を考える必要があるが，これについては後述する．

### 2.2 FEC の適用方法

本論文では，FEC として RS（Reed-Solomon）符号やパリティ符号などのブロック符号を用いることとし，その適用方法を**図 2** に示す．図は受信側での復元の様子を表しており，パケット内の番号はパケットの送出順番を表している．FEC の処理単位を FEC ユニット，複数の FEC ユニットを余分に深さ（$d$）分バッファリングしインタリーブを行う処理単位をインタリーブユニットと定義すると，FEC の回復性能は FEC ユニット内の全パケット数（$N$）に対する FEC パケット数（$N-K$）に依存し，ユニット内で損失箇所が既知の場合，損失パケットが FEC パケット数以下であれば FEC により復元可能である．インタリーブでは，バッファリングした各々のパケットをラウンドロビンにより送出することにより，途中で発生したバースト損失を複数 FEC ユニットに分散させることができ，処理負荷や冗長帯域の増加なしに効率良く損失を復元することが可能になる．

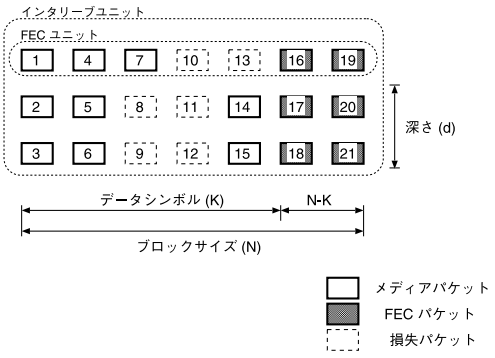


**図 2** FEC の適用方法
Fig. 2 Application method of FEC.

図では，6 パケット連続して損失するバースト損失に対し，$d = 2$ のインタリーブにより 3 つの FEC ユニットに分散して復元する様子を示している．なお本論文では，FEC による回復対象はパケット損失のみとする．パケット内のビット誤りはチェックサムなどによって検出，破棄されることにより，アプリケーション層での FEC 処理時にはパケット損失として処理される．UDP チェックサムは，UDP ヘッダとデータ全体の信頼性を保証し，IPv6 を利用する場合は必須の機能とされている．近年では，一部のビット誤りによるパケット全体の破棄を防ぐために，パケットの任意の先頭部分のみ信頼性を保証する UDP-Lite[17] も提案されている．しかし，本論文ではパケット全体の信頼性を保障するため，通常の UDP チェックサムを用いるものとする．

## 3. 損失パターン特性の決定

ブロードバンドインターネット環境のパケット損失特性を Extended Gilbert モデルで表すために，実ネットワークのトレースデータを採取した．さらに，そのトレースデータより 2.1 節に示した推移行列のパラメータを求め，損失パターン特性を決定する．

### 3.1 実験構成図

本実験における，使用機器および実験構成図を**表 1**，**図 3** に示す．実験では，TCP を含む様々なフローが混在する実運用ネットワーク環境を用いて行った．hostA は広島大学のネットワークに接続された PC で，hostB は FTTH 100 Mbps サービスを受けている一般家庭に設置された PC である．traceroute の結果では，東京を経由して SINET と ISP バックボーンが接続され．両ホスト間には 16 のルータが接続されていた．

### 3.2 測定方法と結果

測定を行うにあたり，RTC（Real Time Clock）を利用して $(i/8192)\,\mu$sec 間隔 $(i = 1, 2, 3, \ldots)$ で RTP/UDP パケットが送出可能なシステムを構築した．これを用いて，hostA から hostB へ一定レート

**表 1** 使用機器の仕様
Table 1 Specification of hosts.

| | CPU | Memory | OS |
|---|---|---|---|
| hostA | P3 Mobile-750 MHz | 256 MB | Vine |
| hostB | P4-1.5 GHz | 512 MB | Linux 2.6 r4 |

のトラフィックを流し，hostB で受信パケットの RTP ヘッダ内のシーケンス番号およびタイムスタンプを記録することで，損失パケットと伝送遅延のトレースを採取した．測定では，データ長 1,128 Byte とし，データ伝送レートは BS/地上デジタル放送のハイビジョン放送波として使用されている MPEG2-TS（Transport Stream）程度を想定し，24.6 Mbps ($i = 3$) とした．

時間帯における平均パケット損失率，平均損失長，損失長の分散，最大損失長とすべてのトレースデータを合わせた統計を**表 2** に示す．このように，特性が時間帯によって大きく変動していることが分かる．ブロードバンド環境下では，伝送帯域が確保されていてもパケット損失が発生し，そのバースト性は時間的な要素により大きく変わるということができる．

### 3.3 パラメータの推定

実トレースデータを基に，損失長と損失間距離の 2 つの観点から損失モデルのパラメータを推定し，評価で用いるロスパターンを決定する．

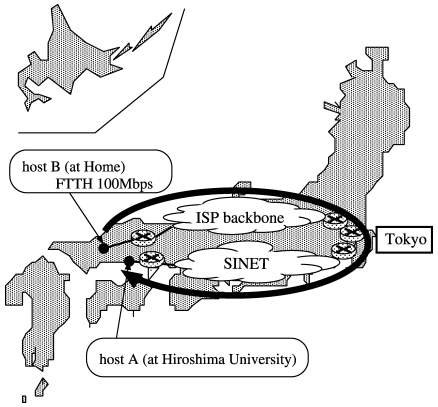


**図 3** 実験構成図
Fig. 3 Experiment environment.

Extended Gilbert モデルの推移確率の計算には，表 2 に示した全トレースデータから得られた損失長の統計を用いた．Extended Gilbert モデルにおける推移確率の計算式は以下のようになる[15]．ここで，$m_i$, $i = 1, 2, ..., n-1$ を長さ $i$ のバースト損失の発生回数（$n-1$ が最大損失長）で，$m_0$ を到着したパケット数とする．

$$p_{01} = \left(\sum_{i=1}^{n-1} m_i\right) \Big/ m_0 \tag{1}$$

$$p_{(k-1)(k)} = \left(\sum_{i=k}^{n-1} m_i\right) \Big/ \left(\sum_{i=k-1}^{n-1} m_i\right) \tag{2}$$

実トレースの特性がモデルに反映されているかどうかを確認するために，算出した推移確率を用いて loss/receive の状態遷移のシミュレーションを行い（試行回数 $10^9$ 回），損失長について記録した．

実トレースと推定したパラメータの比較結果を**表 3** に示す．この結果より，平均値はほぼ実測データに沿って推移しており，バースト損失も実現できていることが確認できる．

次に，採取したトレースデータの損失間距離の統計を**図 4** に示す．横軸はパケット損失間隔で，縦軸は全体に対する発生割合である．先に述べたように，Extended Gilbert モデルは損失長のみで損失発生間隔については考慮されていないが，この図から分かるように，実ネットワークでは損失間距離にかなりの偏りがある．

以上の結果より，FEC の評価を行う際に用いるロ

**表 2** 各時間帯でのトレースデータの詳細
Table 2 Detail of trace data.

| trace | Date / Time / Duration | Avg. Loss Rate [%] | Avg. burst length [pkt.] | deviation [pkt.] | Max. burst length [pkt.] |
|---|---|---|---|---|---|
| 1 | 20040220 / 13:35 - 16:25 / 2 h50 m | 0.00964 | 2.44 | 13.02 | 266 |
| 2 | 20040220 / 16:50 - 19:50 / 3 h | 0.1939 | 1.20 | 0.536 | 23 |
| 3 | 20040221 / 14:45 - 16:25 / 1 h40 m | 0.9681 | 1.24 | 0.576 | 12 |
| 4 | 20040221 / 19:10 - 21:10 / 2 h | 3.9315 | 1.22 | 0.650 | 283 |
| 5 | 20040223 / 10:00 - 12:25 / 2 h25 m | 0.1098 | 1.50 | 3.283 | 193 |
| 6 | 20040223 / 12:30 - 14:50 / 2 h20 m | 0.1686 | 1.39 | 0.790 | 30 |
| 7 | 20040223 / 15:20 - 17:40 / 2 h20 m | 0.1667 | 1.38 | 1.137 | 139 |
| statistics of all traces | | 0.793 | 1.238 | 0.849 | 283 |

**表 3** 実トレースデータと Extended Gilbert モデル
Table 3 Trace data vs. Extended Gilbert Model.

| loss run length | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |
|---|---|---|---|---|---|---|---|---|
| Trace [%] | 81.78 | 14.30 | 2.96 | 0.68 | 0.18 | 0.060 | 0.017 | 0.006 |
| Extended Gibert Model [%] | 81.78 | 14.30 | 2.96 | 0.68 | 0.18 | 0.060 | 0.016 | 0.006 |

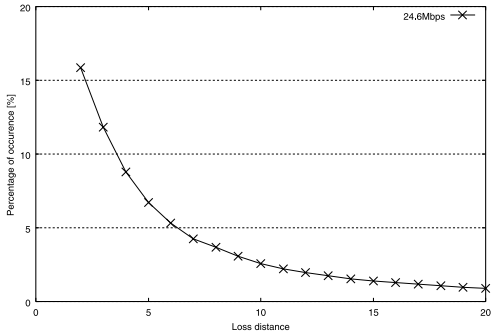


**図 4** 実トレースのパケット損失間距離
Fig. 4 Loss distance of trace data.

スパターンとして，Extended Gilbert モデルによる損失長に加え実トレースで観測された損失間距離の偏りを反映したものを生成する．

## 4. FEC の性能評価実験と適用方法の検討

FEC では演算過程での処理負荷や冗長データを付加することによる帯域増加が生じるため，その適用方法を十分に考慮する必要がある．インタリーブとの併用やブロックサイズを拡大することによりバースト損失を効果的に回復できることが期待できる[8] が，バッファリングによる遅延への影響やその回復性能について定量的な評価を行う必要があると考えられる．本論文では，FEC ソフトウェア処理の基本性能の評価を目的とし，あるロスパターンの下での，回復性能のシミュレーション実験とソフトウェア処理による処理遅延と負荷の実測定より，これらの関係性について調べる．さらに，結果および考察から，広帯域ストリーム伝送における FEC の適用方法について検討する．

### 4.1 実 験 内 容

実験では，FEC として RS 符号を利用し，1 パケット 1,128 Byte，伝送レート 24.6 Mbps の CBR UDP ストリーム（パケット送信間隔は 358.2 $\mu$sec）に対して FEC やインタリーブを利用したときの回復性能と処理負荷について調べた．RS 符号はブロックサイズ $N = 15$，30，60，90 の 4 種類を用いた．実験内容を以下に示す．

#### 4.1.1 回復性能のシミュレーション実験

**図 5** にシミュレーションで想定する環境を示す．FEC のブロックサイズ，冗長度，インタリーブの深さ，および生成したロスパターンを入力パラメータとし，FEC 復元後のメディアパケットの損失率を計算するプログラムを作成し，1 時間のストリーム伝送を想定したときの回復性能について調べた．

FEC の適用方法は図 2 に示すとおりとし，ロスパターンは，Extended Gilbert モデルに図 4 の実トレースの損失間距離を考慮して生成した．その特性を**表 4** に示し，パケット損失率の時間的変動の様子を**図 6** に示す．なお本実験では，冗長度が変化したことによる帯域増加による損失率の変動は考慮していない．回復性能について評価する点は，以下の 2 点である．

(1) 冗長度とインタリーブの深さを変化させた場合の比較
(2) ブロックサイズを大きくした場合とインタリーブの深さを深くした場合の比較

(1) は，インタリーブの深さを変えることでどの程度損失が抑制できるかを調べるためである．(2) は，同じバッファリングパケット数と冗長度（帯域消費量）における回復性能と，同じ誤り訂正能力を提供するときの遅延時間の両側面からの比較を行う．なお，本論文で取り扱う，バッファリングパケット数とブロックサイズ，インタリーブの深さの関係を**表 5** に示す．

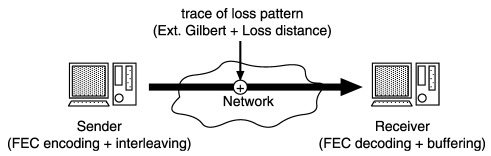


**図 5** シミュレーション方法
Fig. 5 Simulation method.

**表 4** 生成したロスパターンの特性
Table 4 Characteristics of generated loss pattern.

| Avg. Loss Rate [%] | Mean burst length [pkt.] | deviation [pkt.] | Max burst length [pkt.] |
|---|---|---|---|
| 0.6638 | 1.238 | 0.9821 | 139 |

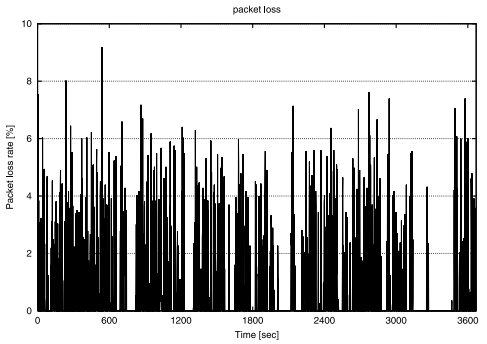


**図 6** パケット損失の時間的変動
Fig. 6 Mean loss rate of generated loss pattern.

#### 4.1.2 FEC 処理負荷の実測定

実トレースデータを採取する際に使用した UDP トラフィック発生プログラムに 1 シンボル 8 ビット長の RS 符号による FEC を実装し，FEC のソフトウェア

処理による負荷を調べた．送信および受信ホストにはCPU PentiumIV-2.8 GHz の PC を用いて測定を行った．RS 符号のソフトウェアには，Phil Karn によって作成された Reed-Solomon CODEC ライブラリバージョン 3.1.1[18] を使用した．

評価の指標は，帯域消費量 1.15 倍固定でシミュレーション実験と同様のブロックサイズでの FEC エンコード/デコード処理による CPU 使用率とした．ここで，CPU 使用率は CPU 総時間 100%からアイドル時間の割合を差し引いたものと定義し，1 時間の平均値とする．デコード処理負荷の測定には，生成したロスパターンに従ってパケット損失を発生させることが可能なロス発生器をホスト間に挿入することで，図 6 のロスパターンで行った．測定結果を**表 6** に示す．

#### 4.1.3 FEC 処理時間の実測定

4.1.2 節と同様の環境で，FEC ユニット単位のエンコード/デコード処理時間について実測より調べた．$10^5$ ユニット分の FEC 処理時間から 1 ユニットあたりの処理時間の平均値を求めた．デコード処理時間の測定には，最も処理時間を必要とする場合として，ユニット内で回復可能な上限値である $N-K$ 個のパケット損失を意図的に発生させ測定した．なお，$N-K$ 個以上のパケット損失が発生した場合には，FEC デコード処理は行われないことに注意が必要である．結果は表 6 に示す．

**表 5** FEC，インタリーブとバッファリングパケット数の関係
Table 5 Relation between FEC, interleaving and buffering size.

| バッファリング | インタリーブの深さ $d$ | | | |
|---|---|---|---|---|
| パケット数 | $(15,13)$ | $(30,26)$ | $(60,52)$ | $(90,78)$ |
| 30 | 1 | 0 | – | – |
| 60 | 3 | 1 | 0 | – |
| 90 | 5 | 2 | 0 | 0 |
| 120 | 7 | 3 | 1 | 0 |
| 180 | 11 | 5 | 2 | 1 |
| 360 | 23 | 11 | 5 | 3 |
| 540 | 35 | 17 | 8 | 5 |

### 4.2 実験結果と考察

4.1.1 節の (1) について，冗長度とインタリーブの深さの変化にともなう回復性能の一例として，$(15,k)$ 符号 ($k=11,12,13$) における測定結果を**図 7** に示す．横軸はインタリーブの深さで，縦軸が FEC 復元後の損失率である．インタリーブユニットを大きくすると，ユニット内に含まれる損失数も増えるために，バッファリング遅延に対する復元性能は低くなる．許容する遅延時間とネットワーク特性から，適切な冗長度と深さを選択する必要があるといえる．

次に，4.1.1 節の (2) および実測定結果より，バッファリングパケット数を同じに設定したとき，ブロックサイズを拡大した場合とインタリーブの深さを変化させた場合の回復性能と処理時間および負荷の比較結果を表 6 に示す．ここで，インタリーブ処理はバッファリングと送出順序を変更する処理のみであり，これによる負荷はないものとする．なお，図中のハイフンで示す部分は，バッファリングパケット数がブロッ

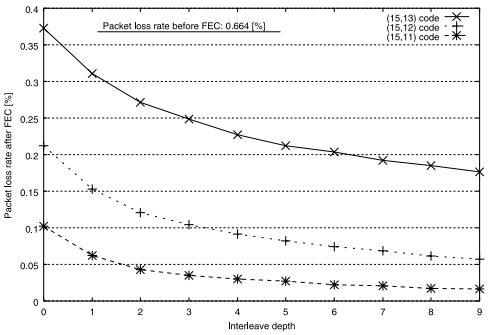


**図 7** インタリーブの効果（N=15）
Fig. 7 Effect of interleaving.

**表 6** インタリーブとブロックサイズを変化させた場合の比較
Table 6 Interleaving vs. expanding block size.

| バッファリング | FEC 復元後の損失率 [%] | | | |
|---|---|---|---|---|
| パケット数 | $(15,13)$ | $(30,26)$ | $(60,52)$ | $(90,78)$ |
| 30 | 0.311 | 0.291 | – | – |
| 60 | 0.248 | 0.216 | 0.189 | – |
| 90 | 0.212 | 0.171 | – | 0.121 |
| 120 | 0.192 | 0.147 | 0.110 | – |
| 180 | 0.170 | 0.118 | 0.081 | 0.063 |
| 360 | 0.135 | 0.081 | 0.041 | 0.027 |
| 540 | 0.119 | 0.064 | 0.025 | 0.017 |
| FEC エンコード処理負荷 [%] | 75.9 | 70.9 | 83.0 | 92.1 |
| FEC デコード処理負荷 [%] | 16.3 | 12.1 | 14.5 | 17.0 |
| FEC エンコード処理時間 [msec] | 3.05 | 5.76 | 14.04 | 24.98 |
| FEC デコード処理時間 [msec] | 4.06 | 7.11 | 16.01 | 27.07 |

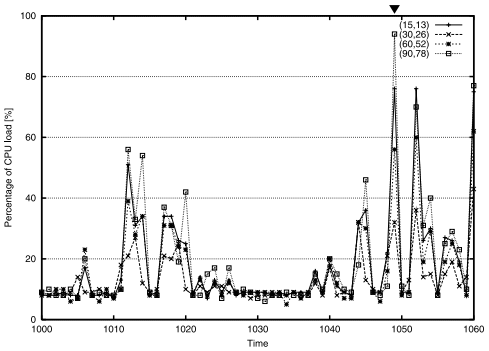


**図 8** デコード処理時の CPU 使用率の時間的推移
Fig. 8 Time-shift of CPU processing load by FEC decoding.

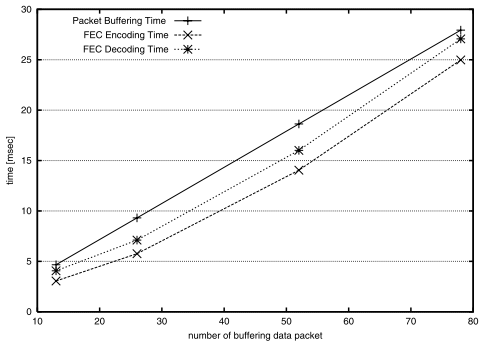


**図 9** FEC 処理時間とパケットバッファリング時間の比較
Fig. 9 Comparison between FEC processing time and packet buffering time.

クサイズ境界と一致しないために，測定不可能な点である．また，**図 8** にデコード処理にともなう CPU 使用率の時間的推移をあわせて示す．

表 6 の結果より，ソフトウェア処理負荷については，エンコード処理ではブロックサイズを拡大することで処理時間および負荷が上昇する傾向にあることが分かる．デコード処理負荷については，図 8 に示すように，細かい粒度で CPU 使用率を見ると，1,049 秒のところ（▼印）などでブロックサイズに応じて CPU 負荷が上昇している部分があることが分かる．しかし，生成したロスパターン特性よりデコード処理を必要としない FEC ユニットが支配的であるため，ブロックサイズによる違いが顕著に現れず，全体的に平均値が低くなっている．また，$(15, 13)$ 符号と $(30, 26)$ 符号を比較すると，$(15, 13)$ 符号の方が負荷が高くなっている．これは，ブロックサイズが小さい範囲ではユニット単位の処理時間がほとんど変わらないため，演算回数の多いことが起因していると考えられる．

ソフトウェア FEC 処理では，高ビットレートのストリームに対して FEC エンコードおよびデコード処理が実時間で処理可能かどうかが重要になる．これは，FEC 処理において，$i$ 番目の FEC ユニットが形成されるまでに $i-1$ 番目の FEC ユニットの処理が終了しているかということと同等であると考えられる．つまり，FEC エンコードおよびデコード処理時間ともに，FEC ユニット形成のためのパケットバッファリング時間より小さい必要がある．FEC 処理時間は，各ブロックサイズにおいて最も処理時間を必要とする場合を考え，表 6 の FEC 処理時間の測定結果から比較した結果を**図 9** に示す．横軸はデータパケットのバッファリング数，縦軸は時間を示す．この結果より，FEC エンコード/デコード処理ともにバッファリング時間を下回っており，実時間で処理可能であることが分かる．

**表 7** FEC とインタリーブによる遅延時間
Table 7 Delay time by using FFC and interleaving.

| バッファリング | 遅延時間 [msec] | | | |
|---|---|---|---|---|
| パケット数 | $(15, 13)$ | $(30, 26)$ | $(60, 52)$ | $(90, 78)$ |
| 30 | 28.79 | 31.5 | – | – |
| 60 | 53.52 | 55.89 | 67.31 | – |
| 90 | 78.25 | 80.28 | – | 107.9 |
| 120 | 102.9 | 104.7 | 118.6 | – |
| 180 | 152.4 | 153.4 | 169.9 | 188.8 |
| 360 | 300.8 | 299.8 | 323.7 | 350.5 |
| 540 | 449.2 | 446.1 | 477.7 | 512.3 |

実測定では，$(90, 78)$ を超えるブロックサイズでは実時間処理が不可能であった．

本論文の構成における，インタリーブと FEC 処理にともなう遅延について考察する．メディアパケットの送出間隔を $i$，FEC エンコード時間を $F_e$，デコード時間を $F_d$，インタリーブの深さを $d$ とし，図 2 における $d = 0$ の FEC ユニットの遅延について求める．なお，本論文では，ソフトウェア処理固有の遅延（メモリの確保，データのコピーなど）についてはブロックサイズやインタリーブの深さに依存しないため考慮しない．

送信側での遅延 $t_s$ は，

$$t_s = \{(K \times i_s) + F_e\}(d + 1) \quad (3)$$

一方，受信側での遅延 $t_r$ は，

$$t_r = \{(N \times i_r)(d + 1)\} + F_d \quad (4)$$

となり，$t = t_s + t_r$ が FEC を付加したことによる遅延となる．

上式と表 6 から求めた遅延時間を**表 7** に示す．なお，送信側の $i_s$ は実験環境と同じく 358.2 $\mu$sec とし，受信側の $i_r$ は帯域消費量 1.15 倍の冗長符号の付加を考慮したレートの 310.5 $\mu$sec とした．この結果より，ブロックサイズが小さなところでは影響がほとんど見られないが，大きくするに従い生成する冗長パケット

も増え，復元パケット数も増加するため，処理遅延時間が増大していることが分かる．ゆえに，ソフトウェア処理の影響としてこの点を注意する必要がある．

最後に，表 6 と表 7 の結果から，ブロックサイズを拡大した場合とインタリーブの深さを変化させた場合の回復性能と遅延について比較を行う．同じ帯域消費量，バッファリング時間であれば，インタリーブユニットよりもブロックサイズを大きくする方が良い回復性能が得られることが分かる．これは，インタリーブでは基本となる FEC ユニットの制約があるのに対し，ブロックサイズを拡大した場合は，FEC ユニット内でのロスパターンに依存することなく回復できるためである．次に，ほぼ同じ誤り訂正能力を提供している FEC 復元後の損失率 0.081 [%] の $(60, 52)$ 符号と $(30, 26)$ 符号について遅延時間を比較すると，インタリーブの深さが深くなる $(30, 26)$ 符号の方が表 7 の遅延時間が大きくなっている．本実験環境では FEC 処理に対して CPU 能力に余裕があるため，ブロックサイズが大きい場合の FEC 処理遅延に比べてインタリーブのためのバッファリング遅延による影響が大きくなっていると考えられる．バッファリング遅延は，バッファリングパケット数とパケットサイズおよび伝送レートによって一意に決まる遅延である．一方，処理時間の測定結果からも分かるように，FEC 処理による遅延は，ブロックサイズに対してリニア以上のオーダで増大し，また CPU 性能によっても大きく変動する．よって，ブロックサイズを大きくすることで CPU 負荷が限界近くの状態になってくると，FEC 処理遅延が増大し，ブロックサイズを大きくする方が遅延時間が大きくなる可能性もある．さらにブロックサイズを大きくすると，リアルタイム処理が不可能となる．インタリーブの利点は，複雑な演算を必要とせず，FEC エンコード，デコード処理遅延および負荷を軽減することができる点にあるといえる．

### 4.3 FEC の適用方法の検討

バッファリング遅延と回復性能はトレードオフの関係にあり，回復性能はネットワーク上のパケット損失特性に大きく依存する．実験結果より，遅延を優先する場合にはブロックサイズを小さくすることが必要であり，回復性能を優先する場合にはインタリーブよりブロックサイズを拡大する手法が適していることが分かる．しかしながら，必要以上に大きなブロックサイズを用いると，バッファリング遅延に加えて FEC 処理遅延の影響を受ける要因となる．ゆえに，ネットワーク特性とアプリケーションの要求条件（遅延や最終的な損失率など）に応じて，ソフトウェア処理で適切なブロックサイズと冗長度を選択し，処理負荷がボトルネックとなる状況下ではインタリーブを用いることで FEC をより効果的に適用することができるといえる．

これらパラメータの決定方法としては，事前測定による決定または受信ノードからのパケット損失率や遅延情報のフィードバックから決定する手法が考えられる．しかし，FEC の適用を想定する環境の 1 つとして，IP マルチキャストでのリアルタイム伝送がある．マルチキャスト環境では，各ノードの接続環境が異なるとネットワーク品質も均一でなくなるため，FEC パラメータの決定が難しくなる．たとえば，ある一端末からの要求に応じて，送信ノードで冗長度を上げる（$N - K$ を増大）ことにより，帯域の狭いネットワークを経由する他の受信ノードで輻輳状態を発生させ，パケット損失を誘発する恐れがある．逆に，冗長度を下げることにより，他の受信ノードで本来回復できるはずのパケットを損失として扱ってしまう可能性が生じる．

このため，IP マルチキャスト環境では複数の受信ノードの要求を考慮した FEC パラメータの決定が必要となるが，この点は今後の課題である．

## 5. おわりに

本論文では，ソフトウェア FEC を用いた損失回復制御の評価について述べた．バースト性を考慮したロスパターンを定義し，そのような損失に対して，ソフトウェア処理に重点を置いた評価を行い，損失回復性能と影響について調べた．その結果，回復性能の点からみると，冗長度を上げることが最も高い復元率が得られ，インタリーブよりブロックサイズを拡大した方が回復性能が良いことが示された．その一方で，遅延時間の点からみると，ブロックサイズを大きくするほど処理遅延と負荷による影響が大きくなることが定量的に示された．また，伝送レートなどから一意に決まるインタリーブによるバッファリング遅延に対して，FEC 処理による遅延は CPU 能力やブロックサイズによって変動するため，環境によってインパクトを与える遅延成分が変わってくることが分かった．特定のモデルにおけるバッファリング遅延と損失回復性能の関係について実測定を用いて調べた結果，数 100 msec の遅延増加を許容することで，実環境で発生するようなバースト的な損失に対しても FEC が効果的に機能することが示された．このことより，アプリケーションの要求条件に応じて適切なブロックサイズ，冗長度を選択する必要があり，インタリーブは，負荷がボトルネックとなる状況下で用いることが適切であるとい

える．

ベストエフォート型のインターネットにおける高ビットレートのストリーム伝送に対して，ソフトウェア処理で損失回復制御により，インタリーブとの併用やブロックサイズの変更などが可能になり，柔軟な損失回復制御が期待できる．また，TCP トラフィックとの公平性を保ちながらストリーム伝送を行う場合でも，TCP-Friendly な帯域制約下でインタリーブやブロックサイズを変更することで，効果的な伝送が実現できると考えている．今後，インターネット上では，多種多様なアプリケーションが利用されることが想定され，また各々の利用形態により，品質に対する要求は多岐にわたると考えられる．ここで示した評価指標は，ソフトウェア FEC を設計する際，基準の 1 つとすることができる．

今後の課題としては，パケット損失の傾向や遅延の要求条件から冗長度，ブロックサイズ，インタリーブの深さの各パラメータを導出するための手法を確立することがあげられる．

## 参 考 文 献


1) 石橋 豊，田坂修二：分散マルチメディアアプリケーションにおけるメディアの時間構造と QoS，電子情報通信会誌，Vol.87，No.3，pp.220–226 (2004).
2) Rizzo, L.: Effective Erasure Codes for Reliable Computer Communication Protocols, *ACM Computer Communication Review*, Vol.24, No.2, pp.24–36 (1997).
3) 近堂 徹，西村浩二，相原玲二，前田香織，大塚玉記：高品質動画像伝送における FEC の性能評価，情報処理学会論文誌，Vol.45, No.1, pp.84–92 (2004).
4) 山内長承，城下輝治，佐野哲央，高橋 修：高信頼同報バルク転送機構，情報処理学会論文誌，Vol.39, No.6, pp.2009–2019 (1998).
5) Rubenstein, D., Kurose, J.F. and Towsley, D.F.: A study of proactive hybrid FEC/ARQ and scalable feedback techniques for reliable, real-time multicast, *Computer Communications*, Vol.24, No.5–6, pp.563–574 (2001).
6) Kostas, T. and Jordan, S.: Packet Erasure FEC on ARQ Protocols, *Proc. SPIE*, Vol.4866, pp.126–137 (2002).
7) Parkins, C., Hodson, O. and Hardman, V.: A Survey of Packet Loss Recovery Techniques for Streaming Audio, *IEEE Network*, Vol.12, No.5, pp.40–48 (1998).
8) Kondo, T., Nishimura, K. and Aibara, R.: Implementation and Evaluation of the Robust High-Quality Video Transfer System on the Broadband Internet, *Proc. International Symposium on Applications and the Internet* (*SAINT*) *2004*, pp.135–141 (2004).
9) Liang, Y.J., G. Apostlopoulos, J. and Girod, B.: Model-Based Delay-Distortion Optimization for Video Streaming Using Packet Interleaving (Nov. 2002).
10) Rizzo, L. and Vicisano, L.: A Reliable Multicast data Distribution Protocol based on software FEC techniques, *Proc. 4th IEEE Workshop on High Performance Communication System*, pp.23–25 (1997).
11) Wu, H., Claypool, M. and Kinicki, R.: A Model for MPEG with Forward Error Correction and TCP-Friendly Bandwidth, *Proc. Workshop on Network and Operating Systems Support for Digital Audio and Video 2003* (2003).
12) 串田高幸，柴田義孝：End-to-End パスにおけるパケット到着間隔および損失の特性解析，情報処理学会論文誌，Vol.44, No.3, pp.570–579 (2003).
13) Borella, M.S. and Swider, D.: Internet Packet Loss: Measurement and Implications for End-to-End QoS, *Proc. Internet Conference oon Parallel Processing*, pp.3–12 (1998).
14) Koodli, R.: One-way Loss Pattern Sample Metrics, RFC3357 (2002).
15) Jiang, W. and Schulzrinne, H.: Modeling of Packet Loss and Delay and Their Effect on Real-Time Multimedia Service Quality, *Proc. NOSSDAV 2000* (2000).
16) Sanneck, H. and Carle, G.: A Framework Model for Packet Loss Metrics Based on Loss Runglength, *Proc. SPIE/ACM SIGMM Multimedia Computing and Networking Conference*, pp.177–187 (2000).
17) Larzon, L.-A., Pink, S., L.-E. Jonsson, E. and G. Fairhurst, E.: The Lightweight User Datagram Protocol (UDP-Lite), RFC3828 (2004).
18) Karn, P.: Reed-Solomon codec 3.1.1. online avaiable at http://www.ka9q.net/code/fec/


(平成 16 年 9 月 6 日受付)
(平成 17 年 9 月 2 日採録)

**近堂　徹**（学生会員）

2001年広島大学工学部第二類（電気系）卒業．2003年同大学大学院工学研究科博士課程前期修了．現在，同大学院工学研究科博士課程後期在学中．IPネットワーク上の高品質動画像伝送，マルチキャスト信頼性保証通信に関する研究に従事．電子情報通信学会会員．

**西村　浩二**（正会員）

1989年広島大学工学部第二類（電気系）卒業．1991年同大学大学院工学研究科博士課程前期修了．全日空システム企画（株）を経て，現在，広島大学情報メディア教育研究センター助手．博士（工学）．マルチメディア機器のリアルタイム遠隔制御，コンピュータネットワークの管理に関する研究に従事．電子情報通信学会会員．

**相原　玲二**（正会員）

1981年広島大学工学部第二類（電気系）卒業．1986年同大学大学院博士課程修了．同大学同学部助手，同大学集積化システム研究センター助教授を経て，現在，同大学情報メディア教育研究センター教授．工学博士．コンピュータネットワークの研究に従事．電子情報通信学会，IEEE Computer Society，IEEE Communication Society各会員．