L-034

誤り訂正符号を用いた トランスポート層プロトコルの実装と評価

A proposal of the transport layer protocol using a forward error correcting

島津圭佑 † 東達軌 ‡ 松澤智史 § 武田正之 §
Keisuke Shimazu Tatsuki Higashi Tomofumi Matsuzawa Masayuki Takeda

1. はじめに

近年インターネットの普及とともに，TV 会議，インターネット TV，オンラインゲームなど，同じデータを同時に多数の端末に送るような通信に対する要求が増加している [1][2]．このような通信では，IP マルチキャスト通信が有用である．

しかし，マルチキャスト通信で各受信端末に対し，ユニキャストと同様にコネクションを張ると，受信端末から届く応答で送信端末に高い負荷がかかる．そのためコネクションレスの通信を用いる場合が多い．

マルチキャスト通信で信頼性を求める方法は大きく分けて 2 つある．1 つ目は応答が少なくなるように工夫したコネクションを張る方法である．2 つ目は信頼性のない通信プロトコル上で FEC(Forward Error Correction) を用いて，受信端末ごとにデータを訂正する方法である．後者の場合，コネクションレスの通信で行えるため，送信端末の処理量は受信端末数に関わらず一定である [3]．

マルチキャスト通信に限らず，一般に FEC を用いた通信をする場合，信頼性のない通信プロトコルを用いる．信頼性のない通信プロトコルである UDP には，ヘッダにペイロードの誤りを検出する目的でチェックサムを格納する領域がある．IPv4 ではこの領域に 0 を入れることでチェックサムを使用しないという選択ができた．しかし，IPv6 ではネットワーク層での誤り検知を行わないため，トランスポート層での誤り検知を必須としている．[4]．チェックサムの計算が合わなかった場合パケット単位で破棄されるため，FEC の復号に用いるデータが少なくなる．このことが原因となり，誤り訂正が失敗することがある．

そこで本稿では，誤りの含まれるパケットを破棄することなく FEC の復号に使用することで，FEC の性能を十分に発揮させるトランスポート層プロトコルを提案し，評価を行う．

2. 既存技術

2.1. 誤り訂正符号

誤り訂正符号は，通信路で生じた誤りを訂正することを目的として，情報データに特定の計算をした冗長データを付加した符号である．このようにして作られた符号を送信し，受信側で誤りを検出する．FEC では，検出された誤りを符号の訂正能力によって訂正する．FEC の利点として，通信路を片方向のみ確保すれば良いことがあげられる．

符号化は，情報データに計算を行い冗長データを作成し，それを情報データに付加し符号語を作成する．復号は，符号語に誤りを含む受信データに計算を行い情報データを算出する．

入力された情報データと冗長データの長さの和を符号長と呼び，符号長に対する情報データの割合を符号化率と呼ぶ．一般的に符号化率が高いほど伝送効率は良いが，訂正能力は下がる傾向がある．誤り訂正符号の一種として，Reed-Solomon(RS) 符号 [5] がある．次元 m では，m ビットのデータを 1 シンボルとして扱い，$GF(2^m)$ 上で符号化，復号を行う．1 シンボル内であれば何ビットの誤りがあったとしても，1 つの誤りとみなし，訂正する．訂正可能量は，付加した冗長データの半分までである．

2.2.FEC Framework

2.2.1. 概要

RFC6363[6] では，アプリケーションと FEC を独立させることを提唱し，その構成要素として，FEC Framework と FEC スキーマを定義している．FEC Framework は UDP や DCCP などのトランスポート層プロトコルと，これらのプロトコル上で動くプロトコルの間に定義される．

FEC Framework を構成する上で用いる FEC スキーマは，データの符号化及び復号を行う機構であり，RFC 5052[7] で使用するものと同様である．FEC Framework では使用する FEC アルゴリズムを受信端末へ伝達する方法についての記載はなく，何らかの方法で受信端末が使用する FEC アルゴリズムを理解しているとして通信を行なっている．

2.2.2.FEC スキーマ

FEC スキーマは，符号化や復号を行う機構である．

符号化の際は，情報データが含まれているパケットと，冗長データが含まれているパケットに分ける必要があり，冗長データを含むパケットが 1 つ以上存在しなければならない．またこの中には，ブロックの番号と，ブロック内での位置を含まなければならない．

復号の際は，何らかの方法で復号器を作成しておき，その復号器に情報データと冗長データを渡すことで復号を行う．復号器を作成する手順については，RFC6363 では定義されておらず，RFC6364[8] などを用いて行う．

†株式会社インターネットイニシアティブ サービスオペレーション本部 サポートセンター カスタマーサポート課, Support Center, Service Operation Division, Internet Initiative Japan Inc.
‡株式会社ネクスト Home's 事業本部 プロダクト開発部 リッテルラボラトリーユニット, Littel Laboratory Unit Product Development Division HOME'S Business Department, NEXT Co., Ltd
§東京理科大学 理工学部 情報科学科, Department of Information Sciences, Tokyo University of Science

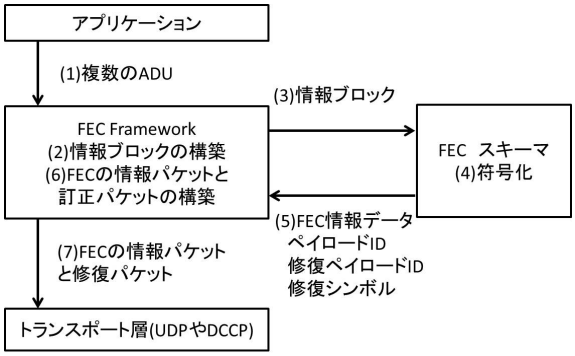

図 1: RFC6363 送信手順

2.2.3. 送受信手順

送信手順を，図 1 に示す．

1. トランスポート層のペイロードとする情報データの 1 単位である ADU(Application Data Unit) を，アプリケーションは FEC Framework へ渡す．
2. FEC Framework は ADU を受け取り，情報ブロックを作成する．
3. FEC Framework は，情報ブロックを FEC スキーマに渡す．
4. FEC スキーマは受け取ったデータを符号化し，冗長データを作成する．
5. FEC スキーマは作成したデータを FEC Framework に渡す．
6. 情報データの場合は，情報データのパケットを構成する．冗長データの場合は，ペイロード ID を付加し，冗長パケットを構成する．
7. 構成したパケットを UDP などのトランスポート層プロトコルで送信する．

受信手順は，図 1 の送信手順を逆に辿ることで行う．

2.2.4. 問題点

既存のトランスポート層プロトコルである UDP は，IPv4 ではチェックサムの使用は任意であるが，IPv6 では使用が必須となった [4]．チェックサムは，パケット内の誤りを検出し，誤りがあった場合パケットを破棄する．図 2 の黒の部分を誤りとして，パケット 1〜6 に対してチェックサムを使用すると全てのパケットは破棄される．受信端末が FEC スキーマで復号する際，破棄されたデータを全て誤りとして認識する．そのため，FEC スキーマで訂正すべき誤りが，FEC の訂正可能量を超えてしまい，復号が失敗するという問題が起こる．

3. 提案手法

3.1. 概要

UDP などの既存のトランスポート層プロトコルを用いると，パケット内に誤りが含まれる場合にそのパケット内に含まれる正しいデータも共に破棄し，FEC が十分に性能を発揮できないという問題を解決するため，本稿では誤り訂正の機構を持ったトランスポート層プロトコルを提案する．

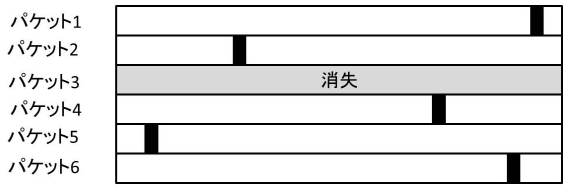

図 2: チェックサムによる問題

提案するプロトコルでは，プロトコルで使用するヘッダに誤りが含まれても，正常に動作する構造を取る．また，データ部では任意の FEC が使用可能である．

3.2. 設計指針

以下に示す指針に従い，プロトコルを提案する．

1. コネクションレスの通信を行う
   FEC を用いた通信を行う状況は，再送困難な場合が多いため，コネクションレスでの通信を行う．
2. 誤りが含まれるパケットを破棄せず，誤り訂正に用いる
   2.2.4 節で述べた問題を解決するために，誤りが含まれているパケットも誤り訂正に使用することで，誤り訂正能力を十分に発揮させる．
3. ヘッダにも誤りが含まれることを考慮する
   ヘッダに誤りが含まれても誤り訂正に支障がない．
4. 任意の FEC を用いることが可能である
   通信の目的や通信状態に合った FEC アルゴリズムを使用できる．また新規の FEC を追加を行える．

3.3. パケットの構成

1 つのパケットは，図 3 に示すように, IP ヘッダ，提案手法の共通ヘッダ，FEC ヘッダ，データという順で構成される．

共通ヘッダとは，提案手法のトランスポート層プロトコルで用いるヘッダのうち，FEC のアルゴリズムによらず必要になる部分である．構成を図 4 に示す．1 つの符号を復号するために必要なデータが含まれるパケットの集まりをブロックと呼ぶ．ブロック開始番号は，ブロックに含まれる最初のパケットのシークエンス番号である．FEC No は，データ部で使用する FEC スキーマの種類を識別するための番号である．ヘッダ長は，FEC ヘッダの長さである．フラグは，3 種類定義されている．

1. 誤りがなければ復号せずに元のデータを取り出せる符号であることを示す．
2. 冗長データが含まれているパケットであることを示す．
3. 最後のブロックであることを示す．

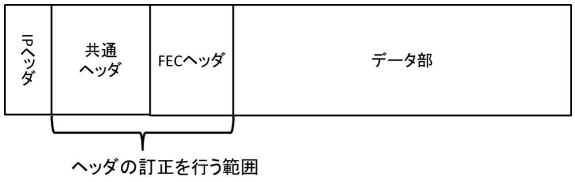

図 3: パケットの構成

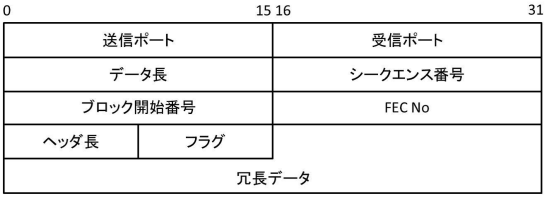

図 4: 共通ヘッダのフォーマット

冗長データは，共通ヘッダの訂正に用いる冗長データである．

3.4.FEC の適応

3.4.1. データ部に用いる FEC

3.2 節の 4 で述べたように，データ部分に対しては，任意の FEC を使用可能とする．そのために，共通ヘッダの FEC No 部分に，FEC のアルゴリズム番号を記載し，復号器の構成に必要な情報は，FEC ごとに独自の FEC ヘッダを作成しそこに記載する．

新たな FEC を実装する場合は，FEC スキーマと FEC ヘッダを作成する．そして，共通ヘッダで FEC スキーマを判別するための番号を振ればよい．この番号は，RFC3452[9] で定義された FEC Encoding ID に相当する．例えば RS 符号の FEC Encoding ID は，1 つのデータ単位 (シンボル) のビット長を定める次元を 2 から 16 の間で選べるものを ID 2，次元 8 のものを ID 5 と定めている [10]．

複数のパケットで FEC を用いた際の構成を図 5 に示す．1 つのパケットには，符号化されたデータの一部が複数個格納される．また，パケット 1 から 6 のように 1 つの復号に使用するパケットの集合をブロックと呼ぶ．

3.4.2. ヘッダの誤り訂正

ヘッダにはポートやシークエンス番号などが含まれており，この部分に誤りがあった場合，他のサービス宛のパケットが届くことや，パケットの順番が分からなくなることなどの問題が起こる．また，FEC 独自のヘッダ部に誤りがあった場合にも，復号方法が分からなくなるなど問題が起こる．そこで，これらのヘッダに対し，FEC を用いて符号化及び復号を行う．ヘッダに対しての FEC の適応は，共通ヘッダと FEC ヘッダのそれぞれで行われる．共通ヘッダの誤り訂正に使用する FEC はアルゴリズムおよびパラメータをプロトコルとして，次元 8，情報データ長 14，冗長データ長 6 の RS 符号を用いるものとする．RS 符号は，1 シンボル内に複数のビットの誤りが発生しても 1 つの誤りとして処理するという特徴があり，シンボル内でバースト的に発生する誤りに強いため用いている．また，ヘッダ長は 32 ビットの倍数が計算機で処理しやすく，ヘッダに 10%の誤りが含まれた場合も対応できるよう，冗長データ長をこのような値とした．一方，FEC ヘッダについては，各 FEC ごとにヘッダを符号化するアルゴリズムと符号化率を定める．

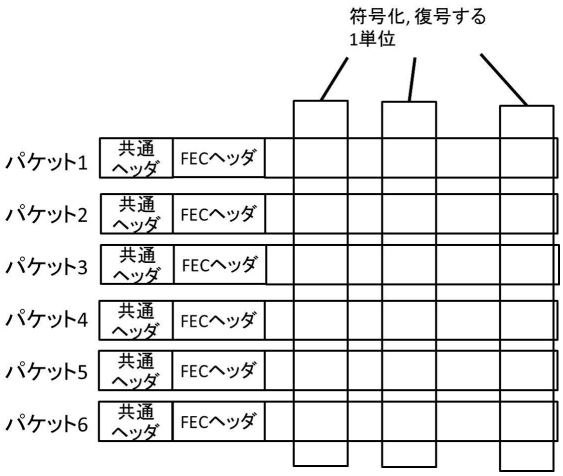

図 5: FEC スキーマの構成

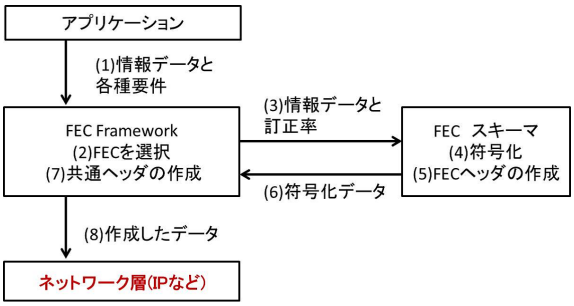

図 6: 提案手法 送信手順

3.5. 送信手順

提案手法の送信手順を図 6 に示す．提案手法の送信で既存手法から大きく変わる点は，トランスポート層のプロトコルであるため，ヘッダを作成する段階でポートなどの指定を行う点と，ヘッダの誤り訂正を行うための冗長データを作成する点である．また，UDP で行われる誤り検出を目的とするチェックサムの計算は提案手法では行わない．

3.6. 受信手順

提案手法の受信手順を図 7 に示す．

既存のトランスポート層で行なっているデータ全体に対してのチェックサムの計算を，ヘッダに対しての FEC に置き換えているため，受信したデータに誤りが含まれる可能性がある．そこで受信手順を次のように変更する．

まず，ネットワーク層からデータを受け取り，共通ヘッ

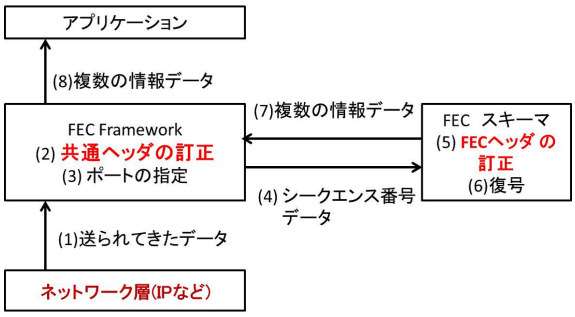

図 7: 提案手法 受信手順

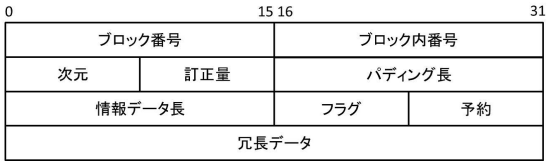

図 8: RS ヘッダのフォーマット

ダの冗長データを用い，共通ヘッダに対して誤り訂正を行う．次に共通ヘッダに書かれたポートを元に，FEC スキーマにデータを渡す．FEC スキーマでは FEC ヘッダに対して誤り訂正を行い，データを保持する．RFC6363 とは異なり，保持するデータに誤りが含まれる可能性がある．同一ブロックのデータが集まったならば，ブロック内番号を元に順番にならべ，誤り訂正を行う．誤り訂正を終えたデータは，ブロック番号を元にブロックの順番を並び替え，FECFramework を経由し，アプリケーションに渡すという手順で受信を行う．

4. 実装

linux-source-3.2.0 を元に，ネットワーク層を IPv6 として，既存手法の UDP 上で FEC を用いた通信を行う RFC6363 と提案手法をそれぞれ実装する．

4.1.FEC スキーマについて

本稿では，RS 符号の FEC スキーマを作成し，実験を行う．

4.2.RS ヘッダとその訂正

既存手法の UDP 上で FEC を用いた通信を行う RFC6363 と提案手法の両方で使用可能な図 8 に示すヘッダを定義する．次元は，1 つのデータ単位 (シンボル) の長さを示す．訂正量は誤りが含まれていても訂正可能な量であり，RS 符号の性質から冗長データの半分の値である．パディング長は，ブロックに含まれるパディングデータの長さである．情報データ長は，データ部の RS 符号に使用する情報データの長さである．フラグは終了ブロックである際に領域に 1 を立てる．提案手法では，冗長データの計算として，情報データ 12，冗長データ 4 の RS 符号を用いた．これは，共通ヘッダと同様の理由である．また，既存手法で使用する場合は冗長データの計算はせず，値を 0 とした．

図 9: 実験環境

表 1: 端末構成

端末名	CPU	クロック数 (MHz)	メモリ (MB)	OS
送信端末	AMD Athlon(TM) Neo Processor	1600	4096	Ubuntu 12.04LTS
中継機	Intel(R) Core (TM) i7 3770S	3100	8192	Ubuntu 12.04LTS
受信端末	Intel(R) Core (TM)2 Duo CPU	3060	4096	Ubuntu 12.04LTS

4.2.1.RS スキーマの復号

復号はピーターソン法 [11] で行う．その際消失したデータ部分に何かしらのデータを入れ，穴埋めが必要となる．そこで全てのビットを立てた値で穴埋めをする．訂正が成功した場合，穴埋めで使用した値による影響は何もない．しかし訂正が失敗した場合，消失したデータの部分には穴埋めで使用した値がそのまま出力される．

1 ブロック分のパケットを集めて復号するため，待ち時間を設定する必要がある．本実装では，待ち時間を同一ブロックのパケットが最後に到着してから 1 秒としている．また，同一ブロックのパケットが全て集まった場合は待ち時間に関わらず復号を開始する．待ち時間を経過しても受信できなかったパケットがある場合には，受信できなかったパケットを消失したデータと見なし，穴埋めをした後，復号を行う．

5. 実験環境と手順

5.1. 実験環境

図 9 のように中継機に，送信端末と受信端末を繋ぎ，IPv6 で通信を行う．

各実験端末の構成を表 1 に示す．

中継機では，パケットを中継する際に任意に誤りを挿入するプログラムを実行する．

誤りの挿入の手順は，まず誤りを挿入するパケットを選択する．次に選択されたパケットに対し，誤りの挿入をする，という手順で行う．誤り挿入は，データリンク層のフレームチェックシークエンス (FCS) を除く部分に対して行われ，誤りが挿入された場合 FCS を再計算が行われる．

以降，誤りを挿入するパケットの選択をする確率を選択率，選択されたパケットに誤りを挿入する割合を挿入率と定義する．誤り挿入で使用するパラメータを 2 つに分けることで，1 パケット内で発生するバースト的な誤りを再現することができる．輻輳が原因で発

表 2: 作成されたデータ

項目名	値
使用したデータ長	1.1MB
1 パケットあたりのデータ長	962 オクテット
ブロック数	14 ブロック
パケット数	1400 パケット
既存手法のパケット長	1040 オクテット
提案手法のパケット長	1052 オクテット

生するバースト的な消失を再現することはできないが，FEC でバースト的な消失を訂正することは難しいため，本稿では実験の対象としない．これらのパラメータを実験時に任意に設定する．

5.2. 実験手順

実験の流れは，次のようになっている．
まず送信端末はデータを符号化し，そのデータをプロトコルに従い送信する．次に，中継機は，送信端末から受け取ったパケットに，誤りを発生させた上で，受信端末に送信する．最後に，受信端末は，受け取ったデータをプロトコルに従い処理し，復号を行う．このとき，時間，受信できたパケット数およびデータの誤り率を計測する．

符号化したデータの送受信で使用するプロトコルを既存手法の UDP 上で FEC を用いた通信を行う RFC6363 と，本稿で提案するプロトコルでそれぞれ実験を行う．また，中継機は選択率，挿入率を任意の値に設定し，誤りを挿入する．

5.3. 使用データ

全ての実験で，データの符号化に次元 $m = 8$，情報データ長 80，訂正可能数 10 の RS 符号を用いた．1.1MB の画像データを符号化し，表 2 に示すデータが作成された．

6. 実験結果

6.1. データの誤り率

入力したデータ量とは，送信端末で符号化に使用するデータの大きさである．残った誤り量は，復号が終わった時点で訂正しきれなかったデータの量であり，入力したデータと訂正後のデータとの差異の量である．誤り率は，受信したデータを復号した結果，訂正されずに残っている誤りの割合のことであり，

$$\text{誤り率} = \frac{\text{残った誤り量}}{\text{入力したデータ量}}$$

である．

1.1MB の画像データに対し，既存手法，提案手法でそれぞれ通信し，誤り率を計測した．

挿入率を 0.1%に固定し，選択率を 0%から 80%までの値に変更して各 10 回計測した結果の平均値を図 10 に示す．既存手法は，選択率 10%で 4.7%の誤りが発生している．選択率を 30%以上では，誤り率が挿入した誤りとほぼ同じ値を示した．一方提案手法では誤り率は全て 0 であった．

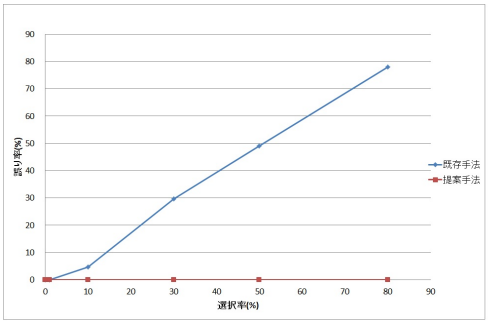

図 10: 挿入率を 0.1%に固定し，選択率を変化させた場合の誤り率

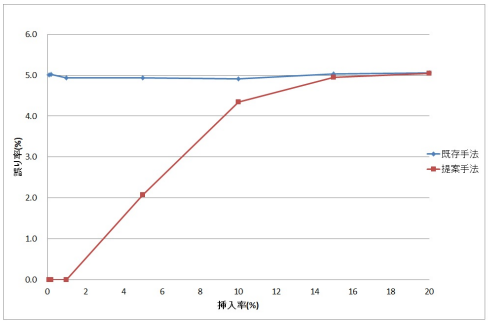

図 11: 選択率を 10%に固定し，挿入率を変化させた場合の誤り率

選択率を 10%に固定し，挿入率を 0.1%から 20%までの値に変更して各 30 回計測した結果の平均値を図 11 に示す．既存手法では，挿入率によらず，5%前後の値となった．また，実験の中で，誤り率が 2%から 9%の範囲の値を得た．一方提案手法は挿入率 1%までは誤り率が 0 であるが，挿入率 5%で 2.1%，15%では既存手法と同程度の 4.9%となった．

6.2. 受信時間

前の実験と同様に，1.1MB の画像データに対し，既存手法，提案手法でそれぞれ通信し，送信端末が送信を始めた瞬間から，受信端末が復号を終えるまでの実時間を計測した．

挿入率を 0.1%に固定し，選択率を 0%から 80%までの値に変更して各 10 回計測した結果の平均値を図 12 に示す．選択率が 1%のとき，既存手法では 2.7sec，提案手法では 1.3sec と 1.4sec の差がある．誤りを挿入しなかった場合を除き，提案手法は既存手法より受信時間が短い．

選択率を 10%に固定し，挿入率を 0.1%から 20%までの値に変更して各 30 回計測した結果の平均値を図 13 に示す．提案手法では，10%まで徐々に復号時間が伸びるという結果を得た．

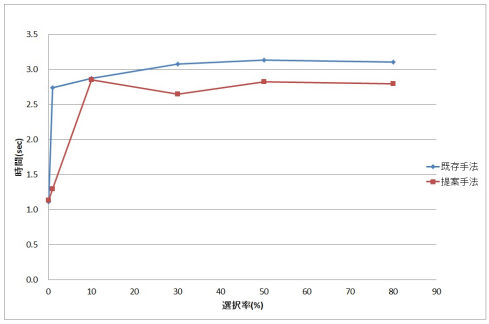

図 12: 挿入率を 0.1%に固定し，選択率を変化させた場合の受信時間

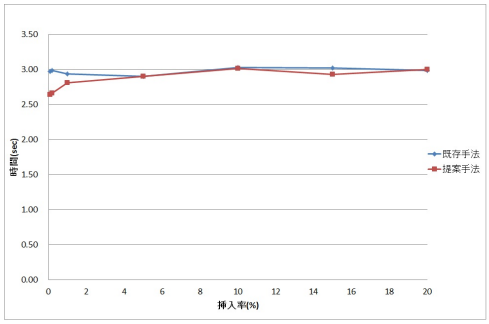

図 13: 選択率を 10%に固定し，挿入率を変化させた場合の受信時間

7. 考察

7.1. データの誤り

提案手法では挿入率を 0.1%に固定した場合は，選択率を変更しても誤り率が 0 のままであった．これは，選択率が 80%のときでさえ，訂正に必要なデータを十分確保できたためである．また，挿入率が 0.1%と低かったため，1 つの符号に復号できなくなるほどの誤りが発生しなかったためである．

既存手法は選択率 10%のとき挿入率に関わらず，約 5%の誤りが発生する．これは，10%の誤りまで訂正可能な符号を用いて実験を行ったため，復号できるブロックとできないブロックが乱数の偏りによって発生した結果，半数の誤り訂正が成功したためである．データのばらつきも，10%まで訂正可能な符号に 10%の確率で誤りを挿入するためである．

既存手法の誤り訂正では，復号の失敗がブロック単位で発生する．提案手法の誤り訂正では，復号の失敗が 1 つの符号のみで起こる場合と，ブロック単位で起こる場合がある．1 つの符号のみで起こる場合は，受信したパケット内に含まれる誤りが原因で発生し，ブロック単位で起こる場合は，パケットが受信できなかったことが原因で発生する．

7.2. 処理時間について

挿入率 0.1%，選択率が 1%のとき，既存手法と提案手法では 1.4sec の差がある．これはパケットの到着を待つ際に，ブロック内のパケット全てが受信できた場合，すぐに復号を開始するよう実装したためである．提案手法では 10 回のうち 9 回で 1400 パケット全てを受信ができたため，パケットの受信待ち時間が短縮され，提案手法が既存手法より 1.4sec 早いという結果が示された．

7.3. ネットワーク機器への影響

ネットワーク層に対して，IPv6 ヘッダの Next Header のフィールドに新たなプロトコル番号を加えるという変更を施した．そのため，IPv6 ヘッダの Next Header フィールドに書かれた未知の値を無視するような機器であれば変更をする必要はないが，破棄する仕様であれば，無視したり，適切な処理をするように設定する必要がある．

一方，トランスポート層で新しいプロトコルを定義しているため，ファイアーウォールや NAT 機器では，設定を変えるなどの対処が必要になる．

8. 今後の課題

本稿では中継機で FCS の再計算を行っているため，FCS の異常によるフレームの欠落は発生していない．そこで，実環境での効果を検証する必要がある．また，実際の通信で起こる誤りの量について，時間帯や環境による影響を調査する必要がある．今回作成したプロトコルを用いる場合，通信をする双方の端末で，既存のプロトコルを置き換えるか，新たなプロトコルとして実装する必要がある．

また，現在の復号状況に応じて FEC の種類や符号化率の変更を要請できるようにする点が挙げられる．

9. おわりに

再送が困難な 1 対多や多対多通信において信頼性を高める手段として，FEC を用いた通信がある．FEC を用いた通信の多くは UDP を利用しているが，IPv6 環境下では UDP のチェックサムの使用が必須とされ，パケット内に誤りが含まれる場合パケット単位で破棄される．受信端末が復号する際，破棄されたデータを全て誤りとして認識する．そのため，訂正すべき誤りが，FEC の訂正可能量を超えてしまい，復号が失敗する．そこで本稿では，誤り訂正機構を持ったトランスポート層プロトコルを提案した．

また，本提案手法を使用した場合と，UDP 上で FEC を使用した場合で比較検証を行った結果，多くのパケットに少しずつ誤りが発生する場合において，本提案手法は既存手法に比べ，誤りの訂正率が高まることが確認された．

参考文献

[1] 総務省，”情報通信白書平成 24 年度版,” 総務省，2012

[2] 中川晋一，”ネットワーク仮想化技術と通信利用型放送,” 情報処理，vol.50，No.11，pp.1106-1109，November 2009

[3] 山本幹, ”信頼性マルチキャスト通信,” 電子情報通信学会誌, vol.85, N0.9, pp-670-674, September 2002

[4] S. Deering, R. Hinden, ”Internet Protocol, Version 6 (IPv6) Specification, ”Request for comments 2460, Internet Engineering Task Force, December 1998

[5] I.S.Reed, G.Solomon, ”Polynomial Codes Over Certain Finite Fields”, Society for Industrial Applied Mathematics, vol.8, No.2, pp.300-304, June 1960

[6] M. Watson, A. Begen, V. Roca, ”Forward Error Correction (FEC) Framework,”Request for comments 6363, Internet Engineering Task Force, October 2011

[7] M. Watson, M.Luby, L. Vicisano, ”Forward Error Correction (FEC) Building Block,”Request for comments 5052, Internet Engineering Task Force, August 2007

[8] A. Begen, ”Session Description Protocol Elements for the Forward Error Correction (FEC) Framework,” Request for comments 6364, Internet Engineering Task Force, October 2011

[9] M. Luby, L. Vicisano, J. Gemmell, L. Rizzo, M. Handley, J. Crowcroft, ”Forward Error Correction (FEC) Building Block, ” Request for comments 3452, Internet Engineering Task Force, December 2002

[10] J. Lacan, V. Roca, J. Peltotalo, S. Peltotalo, ”Reed-Solomon Forward Error Correction (FEC) Schemes, ”Request for comments 5510, Internet Engineering Task Force, April 2009

[11] Daniel Gorenstein, W. Wesley Peterson, Neal Zierler,”Two-error correcting Bose-Chaudhuri codes are quasi-perfect”, INFORMATION AND CONTROL 3, 291-294,September 1960