平成１５年度　研究開発成果報告書
「高速高品質コンテンツ配信を実現する自律適応型メタコンテンツ・ネットワーク技術に関する研究開発」

目　　次

平成１５年度　研究開発成果報告書
「高速高品質コンテンツ配信を実現する自律適応型メタコンテンツ・ネットワーク技術に関する研究開発」

１　研究開発課題の背景

近年、インターネットによる映像配信の気運が高まっているが、乱れのない映像配信を実現するためには、誤り訂正技術が重要な役割を占めている。これまでの誤り訂正技術は、有限体における多項式演算を用いた手法が主流であったが、復号化処理が符号長の自乗に比例して増大するため、専用のハードウェアを用いて処理する必要があった。高速高品質のコンテンツ配信を実現するためには、ソフトウェア的に処理可能な負荷の軽い誤り訂正技術によって、ネットワークの状況(誤り率)に応じて自律的に適応できるメカニズムを導入することが、重要であると考えられる。

２　研究開発分野の現状

符号理論の歴史は、1948 年のシャノンの論文、あるいは 1950 年のハミング符号の発明から始まる。1960 年には、複数のビット誤りを訂正できる BCH (Bose-Chaudhuri-Hocquenghem)符号やバイト誤りを訂正できる Reed-Solomon 符号が発明された。最近、米国 Digital Fountain 社では、新しい誤り訂正符号が開発された。この符号は、バイト誤りを訂正するのではなく、パケット(シンボル)そのものの欠損を補うことの出来る符号であり、これまでの訂正という概念ではなく確率的に元のパケット(シンボル)を復元している。また、この符号は処理時間が符号長に比例するという LDPC (Low Density Parity Check)という誤り訂正符号の流れをくみ、ソフトウェア的に復号可能であるという特徴を持っている。この符号を用いて、自律適応型ネットワークを構成する研究は、有意義であると考える。

３　研究開発の全体計画

3-1　研究開発課題の概要

本研究開発は、最新の超高速低遅延順方向誤り訂正技術を活かした新たな信頼性保証型通信手順によって、従来のフロー制御と廃棄パケットの再送要求やラベルスイッチングによって信頼性を確保するネットワークとはまったく異なる、新たなサービスレベル保証型ネットワークの実現を目指すものであり、基盤技術の研究開発と標準化機関への提案および検証試験により構成される。

3-2　研究開発目標

3-2-1　最終目標

（１）自律適応型のフロー毎アプリケーション毎サービスレベル毎の順方向誤り訂正機能選択制御機能を持つ多段ネットワークのモデル化とシミュレーション技術の確立

（２）自律適応型のフロー毎アプリケーション毎サービスレベル毎の順方向誤り訂正機能選択制御機能を持つ多段ネットワークを実現するためのあらたなIP上の高信頼性コンテンツ配信プロトコルならびにパケット構造案の提唱

（３） 最新の超高速低遅延順方向誤り訂正技術の国際的調査研究と、MPEG-1/2/4等のストリーミング配信やダウンロード配信におけるネットワーク上での多段処理（エンコード・デコード・トランスコード）に適した同技術のクライテリアの確立

（４） デファクト標準化を念頭においた、順方向誤り訂正機能およびその選択制御機能のデバイス化と試作評価プラットフォームとしてのサーバ・クライアントおよびホームゲートウエイ等低速低価格中継装置の試作ならびに実証試験

（５） 実証試験結果に基づく、ＩＥＴＦ等への自律適応型のフロー毎アプリケーション毎サービスレベル毎の順方向誤り訂正機能選択制御機能を持つ多段ネットワークを実現するためのあらたな高信頼性コンテンツ配信プロトコルのドラフト提案

（６） 本研究の信頼性保証型通信手順が、インターネット世界に受け入れられるための、適切な抑制手段に関する研究と提唱。

（７） 中継装置に適した超高速低遅延の順方向誤り訂正アルゴリズムの調査研究とデバイス試作。

（８） キャリア・ＩＳＰ等異業種の参画を得たフィールド実験と有用性の検証

3-2-2　中間目標

（１） MPEG1/2/4等のストリーミング配信やダウンロード配信におけるネットワーク上での処理（エンコード・デコード・トランスコード）に適した順方向誤り訂正機能のクライテリアの明確化と、これに適した最新の超高速低遅延順方向誤り訂正技術ならびに実用化課題に関する調査研究をおこなう。

最新の超高速低遅延順方向誤り訂正技術の調査研究と動向を把握し、本研究開発におけるネットワーク上の利用に適した１つまたは複数の順方向誤り訂正技術候補の選択を行なう。

ネットワーク上での実用化のための課題（ハードウェアを用いた高速化の必要性、チューニングパラメータ、トレードオフの関係にある特性等）について研究する。更に、独自の超高速低遅延順方向誤り訂正技術の研究開発の可能性について検討を行なう。

（２） 自律適応型のフロー毎アプリケーション毎サービスレベル毎の順方向誤り訂正機能選択制御機能を持つ多段ネットワークのモデル化を行なう。

自律適応型のフロー毎アプリケーション毎サービスレベル毎の順方向誤り訂正機能選択制御機能を持つ多段ネットワークのモデル化を行なう。

シミュレーション技術の調査研究を行ない、必要に応じてシミュレーションプログラムの試作を行なう。

（３） 自律適応型のフロー毎アプリケーション毎サービスレベル毎の順方向誤り訂正機能選択制御機能を持つ多段ネットワークを実現するためのあらたなIPプロトコルならびにパケット構造案の素案を作成する。

自律適応型のフロー毎アプリケーション毎サービスレベル毎の順方向誤り訂正機能選択制御機能を持つ多段ネットワークを実現するためのあらたなIPプロトコルならびにパケット構造のクライテリアに関する調査研究を行なう。

上記クライテリアと既存のプロトコル提案との関連について調査研究を行なう。

（4）デファクト標準化を念頭においた、順方向誤り訂正機能およびその選択制御機能のデバイス化と評価プラットフォームとしてのサーバ・クライアントおよびホームゲートウエイ等低速低価格中継装置の試作を行なう。

3-3　研究開発の年度別計画

研究開発項目	１４年度	１５年度	１６年度	備考
1）　ネットワークのモデル化[①]とシミュレーション技術の開発[②]ならびに高信頼性コンテンツ配信プロトコルの研究[③]	①-----→ ②------ ③------	 -------→ ----------	 -------→	①②③の一部を大阪大学に再委託
2）　最新の高速低遅延順方向誤り訂正技術の国際的調査研究[④]とネットワーク上の多段利用を想定したクライテリアの確立[⑤]	④-----→ ⑤------	---→		④⑤の一部を京都大学に再委託
3）　デバイス化[⑥]とホームゲートウエイ等評価プラットフォームの試作[⑦]、実証試験[⑧]およびＩＥＴＦ等へのドラフト提案[⑨]	⑥------ ⑦------	→ ----→ ⑧--------- ⑨--	 → ----------→	
4）　帯域抑制手段に関する研究および普及に向けた補完技術に関する研究[⑩]、並びに既存の中継装置を補完する高速高品質コンテンツ配信用装置に関する研究と試作[⑪]		⑩------ ⑪	----------→ ----------→	

４　研究開発の概要

4-1　研究開発実施計画

4-1-1　研究開発の計画内容

（A）最新の高速低遅延順方向誤り訂正技術の国際的調査研究とネットワーク上の多段利用を想定したクライテリアの確立

インターネット上での、大容量の映像・音声・音楽・データの高速かつ遅延の少ない配信に適した誤り訂正技術の国際的調査研究と、サーバ・クライアント・中継装置を想定したOn_the_flyでの順方向誤り訂正エンコード・デコード・トランスコード処理に求められるクライテリアについて整理する。

（１）最新の超高速低遅延順方向誤り訂正技術に関する国際的調査研究

高速低遅延順方向誤り訂正技術に関する最新の国際的研究動向を把握し、本研究開発におけるネットワーク上の利用に適した１つまたは複数の順方向誤り訂正技術候補の選択と、実用化のための課題（ハードウェアを用いた高速化の必要性、チューニングパラメータ、トレードオフの関係にある特性等）について研究する。更に、独自の超高速低遅延順方向誤り訂正技術の研究開発の可能性について検討を行う。

（２）ネットワーク上で多段利用可能な順方向誤り訂正技術に関するクライテリアの分析

所定のアプリケーションに対して、フロー毎に、サービスレベルに応じて、選択的に、自律適応的に順方向誤り訂正技術適用する多段ネットワークの実現に適した、順方向誤り訂正技術のクライテリアの調査研究を行なう。

（B）ネットワークのモデル化とシミュレーション技術の開発ならびに高信頼性コンテンツ配信プロトコルの研究

自律適応型のフロー毎アプリケーション毎サービスレベル毎の順方向誤り訂正機能選択制御機能を持つ多段ネットワークのモデル化を行なう

（１）ネットワークのモデル化

所定のアプリケーションに対して、フロー毎にサービスレベルに応じて順方向誤り訂正機能の選択制御機能を持つ多段ネットワークのモデル化を行なう。

（２）シミュレーション技術の研究開発

メタコンテンツ型の順方向誤り訂正技術を、所定のアプリケーションに対して、フロー毎にサービスレベルに応じて適用するネットワーク・モデルにおけるＱｏＳのシミュレーション技術の研究開発と検証を行なう。

（３）高信頼性コンテンツ配信プロトコルの研究開発

ブロードバンドコンテンツのライブあるいはオンデマンド配信サービスを提供するネットワークシステムにおいて、順方向誤り訂正機能を有さないサーバ、順方向誤り訂正機能を有するサーバ、順方向誤り訂正機

能を有さないクライアント、および、順方向誤り訂正機能を有するクライアントが混在する可能性のある、ヘテロジーニアスなネットワーク環境下において、所定のアプリケーションに対してフロー毎に、サービスレベルに応じた順方向誤り訂正機能を自律適応的に、あるいは、利用者の要求に応じて、選択的に適用・制御するためのプロトコル（パケット構造および通信シーケンス）案に関する、研究・開発をおこなう。

（C）デバイス化とホームゲートウエイ等評価プラットフォームの試作、実証試験およびＩＥＴＦ等へのドラフト提案

（1） 順方向誤り訂正機能のデバイス化

順方向誤り訂正機能およびその選択制御機能のデバイス化(プロトタイプ試作)をおこなう。デバイス化は、（2）の①項の検討結果に基づき当該順方向誤り訂正技術の特徴を生かし、かつ、（2）の②項のクライテリアの調査研究結果を参考に、なるべく可用性が高く、汎用的、経済的な方法でおこなう。

（2） 評価プラットフォームの試作開発

フロー毎アプリケーション毎サービスレベル毎の順方向誤り訂正技術の必要性・有効性と上記プロトコルの実現性・妥当性を検証するため、順方向誤り訂正機能およびその選択制御機能を搭載した評価プラットフォームとして、下記の実証装置の試作を行なう。

a) サーバ・クライアント間のネットワークに挿入して、既存の順方向誤り訂正機能を有さないサーバから複数のクライアントに提供される複数のストリーミング・コンテンツに対し、フロー毎に所定のアプリケーションに対して、サービスレベル契約に応じたＩＰパケットの順方向誤り訂正符号化をリアルタイムに実施する、サーバ外付用高速ＦＥＣエンジン。

b) 所定のアプリケーションに対するフロー毎のＩＰパケットの順方向誤り訂正復号化機能およびサービスレベルのモニタリング機能を有するクライアントとしてのＳＴＢ(セットトップボックス)またはホームゲートウエイ。

c) 所定のアプリケーションに対するフロー毎のＩＰパケットの順方向誤り訂正符号化および復号化機能、ならびに、サービスレベルのモニタリング機能を有するＰ２Ｐサーバ・クライアントとしての双方向ＳＴＢ(セットトップボックス)またはホームゲートウエイ。

（3） 標準化提案の為の実証試験

フロー毎アプリケーション毎の順方向誤り訂正技術の必要性・有効性と、上記プロトコルの妥当性を実証するため、国内の代表的な、通信キャリア、ブロードバンドサービスプロバイダ、および、インターネット総合研究機関と以下の２種類のフィールド実験を行なう。

a) Ｂ２Ｃモデル

順方向誤り訂正機能を有さない一般的なＶＯＤまたはＬＩＶＥサーバ、所定のアプリケーションに対してフロー毎にサービスレベル

契約に応じたＩＰパケットの順方向誤り訂正符号化をリアルタイムに実施できるサーバ外付型高速ＦＥＣエンジン、および、所定のアプリケーションに対してフロー毎のＩＰパケットの順方向誤り訂正復号化機能とサービスレベルのモニタリング機能を有するクライアントとしてのＳＴＢ(セットトップボックス)を用いた、Ｂ２Ｃ型サービスにおける順方向誤り訂正技術の必要性・有用性・妥当性の実証試験。

b) Ｐ２Ｐモデル

所定のアプリケーションに対してフロー毎のＩＰパケットの順方向誤り訂正符号化および復号化機能、ならびに、サービスレベルのモニタリング機能を有する、サーバ・クライアントとしての双方向ＳＴＢ(セットトップボックス)またはホームゲートウエイを用いた、Ｐ２Ｐ型サービスにおける順方向誤り訂正技術の必要性・有用性・妥当性の実証試験。

（4） ＩＥＴＦ等標準化団体へのドラフト提案

「所定のアプリケーションに対するフロー毎のサービスレベル契約に応じた順方向誤り訂正機能の選択制御機能を持つネットワークを実現するためのプロトコルに関する検討結果」、「シミュレーション結果」、ならびに、「評価プラットフォームを用いたネットワークのサービスレベルのモニター結果、および、順方向誤り訂正技術の実証試験結果」を踏まえ、国内外の適切なデファクト標準化組織に対し、ブロードバンドコンテンツのライブあるいはオンデマンド配信サービスにおける、順方向誤り訂正技術の必要な領域、有用な範囲、および、順方向誤り訂正機能の選択制御機能を持つネットワークを実現するためのプロトコルに関するドラフト提案の作成を行なう。具体的には、国内のデファクト標準化組織として、ＨＳＡＣ（光サービスアーキテクチャコンソーシアム）の後継検討会である光アプライアンスインタフェース検討会等を、国際的デファクト標準化組織としてＩＥＴＦ (Internet Engineering Task Force)等を提案の対象とする。

（D）帯域抑制手段に関する研究および普及に向けた補完技術に関する研究

本研究の信頼性保証型通信手順がインターネット社会に受け入れられるための適切な帯域抑制手段、および、評価プラットフォームを用いた実証試験結果の分析に基づく本研究技術の普及に向けた課題と、課題解決に重要な補完技術、に関する調査研究を行なう。

4-2 研究開発の実施内容

（A） 誤り訂正技術に関する最新の国際的な研究動向を調査し、その中から Reed-Solomon 符号を取り上げた。Digital Fountain (DF) 社の LT 符号ならびに Raptor 符号(Multi-Stage 符号)と、Reed-Solomon 符号に関して、アルゴリズム特性ならびに欠損復元能力の比較検討を行った。

LT 符号は、Low-Density Parity Check 符号の流れをくむ符号であり、復号のアルゴリズムが符号長の一乗に比例する、つまり $O(n)$ であることを特徴としている。現在、広く用いられている Reed-Solomon 符号は、符

号長の自乗に比例する $O(n^2)$ であり、復号時間を短縮するために専用ハードウェアを使用している。汎用 CPU でソフトウェア的にデコード処理ができることが、LT Coding の大きな利点である。ソフトウェア的に処理できるため、広範囲のネットワーク誤り率に自律的に適応することが可能で、今回の研究テーマに対応できる符号であると考えられる。

LDPC(Low-Density Parity-Check)符号は、シャノンの限界に迫る符号として最近注目を集めている。LT Coding の改良型である Raptor 符号は、LT 符号,LDPC 符号,拡大 Hamming 符号の３つを組み合わせた符号であり、Multi-Stage 符号とも呼ばれている。他の２つの符号と組み合わせた結果、LT 符号において 10^{-8} 程度であった復元失敗率が、Raptor 符号では 10^{-12} 以下に改良されている。

誤り訂正符号では、バースト状の誤りに対して、符号長を長くする等の方法を用いて対応することが可能である。ただ Reed-Solomon 符号では、符号長を長くすると、デコードにかかる時間が $O(n^2)$ であるため、実用的ではない。上記の点を数値計算で比較し、IP ネットワークにおけるパケット欠損の修復において、Digital Fountain 社 LT 符号の改良型である Raptor 符号が、有望であるという目処を得ている。

この符号を用いたシステムを設計するために、トレードオフの関係にあるバースト耐性(符号長)ならびに遅延時間(プロテクション時間)に関する検討を行った。これらのパラメータは複雑に関係しており、これらの関係をまとめることは非常に有益である。

（Ｂ） Raptor符号のアルゴリズムの優秀性を比較するために、Reed-Solomon符号と処理速度面での比較を行った。実機でReed-Solomon符号を走らせたところ、符号長の自乗にほぼ比例して、処理時間が長くなることが分かった。Raptor符号は符号長の処理時間の一乗に比例するため、Reed-Solomon符号より処理時間が短いことが分かった。

インターネット上で映像配信サービスを行うためには、送信途中に発生する欠損パケットを復元することが重要である。欠損パケットを復元する方法としては、ARQ (Automatic Repeat reQuest)と呼ばれる再送方式や、Reed-Solomon符号を基にした欠損復元符号が用いられている。しかし、再送による遅延時間の増加や、冗長データを付加することによる処理時間の増加が問題となっており、また回線ごとに異なるパケットロスへの対処が課題となっている。これら問題解決のために米国Digital Fountain社のRaptor符号等を選択し、回線ごとに異なるパケット欠損に対応するサポートサーバを設置して、欠損の多い回線に対して冗長パケットをさらに追加する方法を考案し、送信完了時間が約15%減少することを確認した。本研究は、大阪大学への委託研究の成果であり、電子情報通信学会研究会で発表した。

誤り訂正(欠損復元)符号を実際のインターネットに適用する場合、インターネットのパケット欠損率やジッタを知ることは重要である。今回、インターネットに接続された折り返し回線を用いて、パケットの出発時間と到着時間との関係を測定することによって、ジッタを測定し、ジッ

タとパケット欠損の関係をまとめた。単純なモデルでは、パケットの到着時間が遅れると、パケット欠損が多発することになるが、実験結果では必ずしも、このモデルと一致しないため、再検討が必要である。

今後、これらのモデルにプロトコルを組み込んだ形での改良を行い、最終形態である自律適応型のフロー毎アプリケーション毎サービスレベル毎に必要な誤り訂正方法の検討を行う。

（C） サーバ・クライアント間のネットワークに挿入して、欠損復元機能を有さないサーバが送出する複数のストリーミング・コンテンツに対し、フロー毎のサービスレベル契約に応じて、欠損復元機能を付与する高速FEC(Forward Error Correction)エンジンを開発した。

上記高速ＦＥＣエンジンを経由したフロー毎に符号化されたコンテンツに対して、サービスレベルのモニタリング機能を有するクライアントとしてＳＴＢを用いて、FTTH ユーザ及び ADSL ユーザに対して FEC の有効性を確認する実証実験を行った。

実験期間は、2003 年 12 月 8 日(月)～2004 年 1 月 31 日(土)までの約２カ月で、モニタは東京都内に住む約 300 名を対象とした。コンテンツ数は、映画・アニメ・音楽等の約 100 本である。定量的(客観)データとして、STB にＱｏＳ評価用エージェントを実装し、コンテンツ視聴毎にＱｏＳデータおよびＦＥＣ機能の運用データ(パケットロス等)を収集した。定性的(主観)データとして、コンテンツ視聴毎に画質，音声品質，操作性，応答性に対するアンケートを画面表示し、STB 付属のリモコンで３段階評価(良い、普通、悪い)のアンケートを実施した。その結果、パケットロスが発生した場合に FEC を用いないとユーザの主観評価が悪く、視聴時間・回数ともに減少することがわかり、FEC は安定したサービス供給をするのには不可欠な技術であることを実証した。

さらに順方向誤り訂正技術の必要な領域、有用な範囲、および順方向誤り訂正機能の選択制御機能を持つネットワークを実現するためのプロトコルを設計・実装し、上記の実証実験において実際に使用し、有効性を確認した。そして、そのプロトコルを IETF へ提案する準備として、本テーマに関連する IETF における標準化動向の調査検討、および、ドラフト案の作成を行った。調査検討の結果、下記のワーキンググループのいずれかを提案先とする予定である。

・AVT(Audio/Video Transport)：

RTP(Real-time Transport Protocol)(音声、画像データを実際に転送するプロトコル)関連

・MMUSIC(Multiparty Multimedia Session Control)：

RTSP(Real Time Streaming Protocol)(音声、画像データの転送方法を制御するプロトコル)関連

・RMT(Reliable Multicast Transport)：

FEC(Forward Error Correction)(順方向誤り訂正機能)関連

（D） 本研究で使用している信頼性保証型通信手順がインターネット社会に受け入れられるために、UDP プロトコルによる通信帯域が、正帰還により急激に増大することのない帯域抑制手段を検討中である。さらに、評価プラットフォームを用いた実証試験結果の分析を継続中である。

５　研究開発実施状況

5-1　最新の高速低遅延順方向誤り訂正技術の国際的調査研究とネットワーク上の多段利用を想定したクライテリアの確立

5-1-1　序論

インターネット上で、MPEG1/2/4、WMT、Real、QT、MP3等のファーマットの映像・音声・音楽のストリーミング配信や大容量ファイルの高速ダウンロード配信を行なう場におけるサーバ・クライアント・中継装置を想定したOn_the_flyでの順方向誤り訂正エンコード・デコード・トランスコード処理に求められるクライテリアについて整理する。

高速低遅延順方向誤り訂正技術に関する最新の国際的研究動向を把握し、本研究開発におけるネットワーク上の利用に適した１つまたは複数の順方向誤り訂正技術候補の選択と、実用化のための課題（ハードウェアを用いた高速化の必要性、チューニングパラメータ、トレードオフの関係にある特性等）について研究する。更に、独自の高速低遅延順方向誤り訂正技術の研究開発の可能性について検討を行なう。

5-1-2　研究内容

誤り訂正符号として、Reed-Solomon 符号を取り上げ、この符号と米国 Digital Fountain 社 Dr. Luby らにより開発された LT 符号について評価検討を行った。LT 符号は、Low-Density Parity-Check (LDPC) 符号の流れをくむ符号であり、復号のアルゴリズムが符号長の一乗に比例する、つまり $O(n)$であることを特徴としている。現在、広く用いられている Reed-Solomon 符号は、符号長の自乗に比例する $O(n^2)$であり、復号時間を短縮するために専用ハードウェアを使用している。汎用 CPU でソフトウェア的に復号処理ができることが、LT 符号の大きな利点である。ソフトウェア的に処理できるため、広範囲のパケット欠損率に適用することが可能で、今回の研究テーマに対応できる符号であると考えられる。

LDPC (Low-Density Parity-Check) 符号は、シャノンの限界に迫る符号として最近注目を集めている。LT Coding の改良型である Raptor 符号は、LT 符号，LDPC 符号，拡大 Hamming 符号の３つを組み合わせた符号であり、Multi-Stage 符号とも呼ばれている。他の２つの符号と組み合わせた結果、LT 符号において 10^{-8} 程度であった復元失敗率が、Raptor 符号では 10^{-12} 以下に改良されている。

誤り訂正符号では、バースト状の誤りに対して、符号長を長くする等の方法を用いて対応することが可能である。ただ Reed-Solomon 符号では、符号長を長くすると、デコードにかかる時間が $O(n^2)$であるため、実用的ではない。上記の点を数値計算で比較し、IP ネットワークにおけるパケット欠損の修復において、Digital Fountain 社 LT 符号の改良型である Raptor 符号が、有望であるという目処を得ている。

さらにネットワーク上での(実用化のための)トレードオフに関する研究として、Raptor符号において、ネットワーク上のパケット欠損として二項分布を仮定し、帯域オーバヘッドを考慮した誤り訂正能力と遅延時間に関する検討を行った。

なお本研究に関して、基礎的な部分は京都大学・高橋研究室に再委託を行っ

た。再委託の研究成果報告書は、本報告書の最後に添付する。

5-1-3　誤り訂正符号の基本的な考え方

符号理論の歴史は、1948 年のシャノンの論文、あるいは 1950 年のハミング符号の発明から始まる。1960 年には、複数のビット誤りを訂正できる BCH (Bose-Chaudhuri-Hocquenghem)符号やバイト誤りを訂正できる Reed-Solomon 符号が発明された。今回取り上げる Digital Fountain 社の誤り訂正符号(LT Codes ならびに、その改良型である Raptor codes)は、パケット(シンボル)そのものの欠損を補うことの出来る符号であり、これまでの訂正という概念ではなく確率的に元のパケット(シンボル)を復元できるという方法である。図 5-1-3-1 に、誤り訂正符号の歴史を示す。

表 5-1-3-1　誤り訂正符号の歴史

<table>
<tr><th></th><th>符号化法</th><th>復号化法</th></tr>
<tr><td rowspan="4">Reed
-Solomon
符号</td><td rowspan="4">Reed-Solomon法：O(n^2)
(1960年)</td><td>Peterson法：O(n^3) (1961年)</td></tr>
<tr><td>Berlekamp-Massey法：O(n^2)
(1968年)</td></tr>
<tr><td>Euclid法：O(n^2) (1975年)
(杉山, 平澤, 笠原, 滑川)</td></tr>
<tr><td>Yule-Walker法：O(n^3)</td></tr>
<tr><td>LDPC符号</td><td>Richardson法：O(n^2)
(2001年)</td><td>Sum-product法：O(n)
(Gallager)　(1962年)</td></tr>
<tr><td>LT符号
Raptor符号</td><td>ランダム組合せシンボルに
XOR演算：O(n) (2001年頃)
(Luby & Shokrollahi)</td><td>専用アルゴリズムでの
解法：O(n) (2001年頃)
(Luby & Shokrollahi)</td></tr>
</table>

LDPC：Low-Density Parity-Check

5-1-3-1　Forward Error Correction (FEC)

伝送中の欠損パケット(シンボル)を補う方法としては、以下の２つに分類される。

（１）ARQ (Automatic Repeat reQuest)

（２）FEC (Forward Error Correction)

ARQ (Automatic Repeat reQuest)は、送信側に欠損した情報の再送を要求することによって欠損パケットを補う方法である。これに対して Forward Error Correction (FEC)方式は、その名前が示す様に(再送要求無しで)前方で欠損を補う方法である。FEC 方式では、送信側で冗長なコードを付加することによって、伝送路でビットエラーやパケット(シンボル)欠損が発生しても、受信側で損失したパケット(シンボル)を回復できる方法である。ARQ 方式では、再送要求によるパケットが到着するまでに遅延時間 (RTT : Round Trip Time) が発生するが、

FEC 方式では、RTT による遅延が発生しない。一方、本来の情報部分に冗長部分を付与する必要があるため、余分にデータを送信する必要がある。このため「受信時間が長くなる」「余分な伝送帯域が必要となる」などの問題が発生する。また、想定数以上のパケットが欠損すると、復元が不可能になるという問題もある。

図 5-1-3-1 は、同じ符号を３回送るという最も簡単な誤り訂正符号を示した物である。「0」を送る場合には、「000」を、「1」を送る場合には、「111」を送ることになる。３次元空間で考えると、座標(0,0,0)から(1,1,1)に移動するためには、３本の辺を通過しなければならない。従ってこの符号の最小 Hamming 距離は、3 となる。

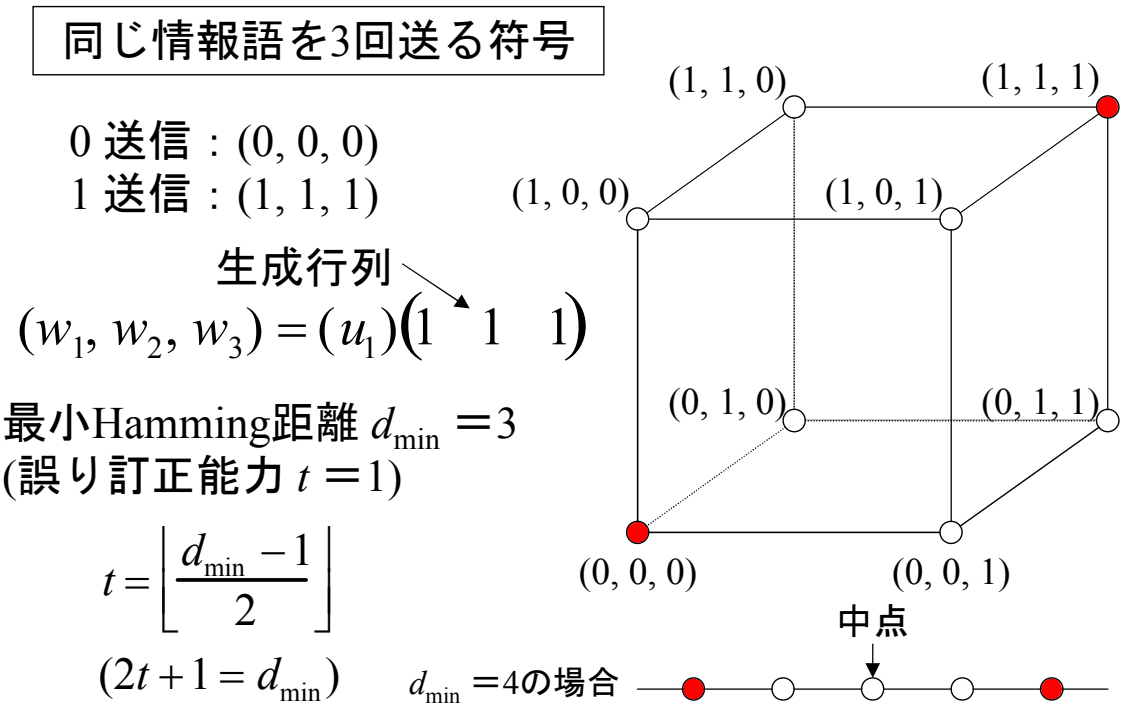


図 5-1-3-1　最小 Hamming 距離

図 5-1-3-2 は、誤り訂正符号の能力を示している。このうち 1 つに誤りが発生し、「100」となった場合、３次元空間で考えると、本来の座標より距離が 1 だけ離れた位置(1,0,0)になる。３次元座標の中で許されるのは、(0,0,0)と(1,1,1)の２点であるため、符号「100」は(0,0,0)に近く、確率的には「000」であると考えるのが妥当である。このようにして、誤りを検出する方法を、限界距離復号法と呼ぶ。同様に符号「101」を受信した場合には、「111」が送信されたと考えるのが妥当である。また、２個間違った場合には、別の符号語と判断される。つまり、「0」を送信した際に、「110」と誤った場合には、受信側では「1」が送信されたと理解され、誤訂正となる。この符号は、符号長 n=3, 情報長 k=1, 誤り訂正能力 t=1 の符号となり、この報告書では、(n, k, t)誤り訂正符号と表現する。

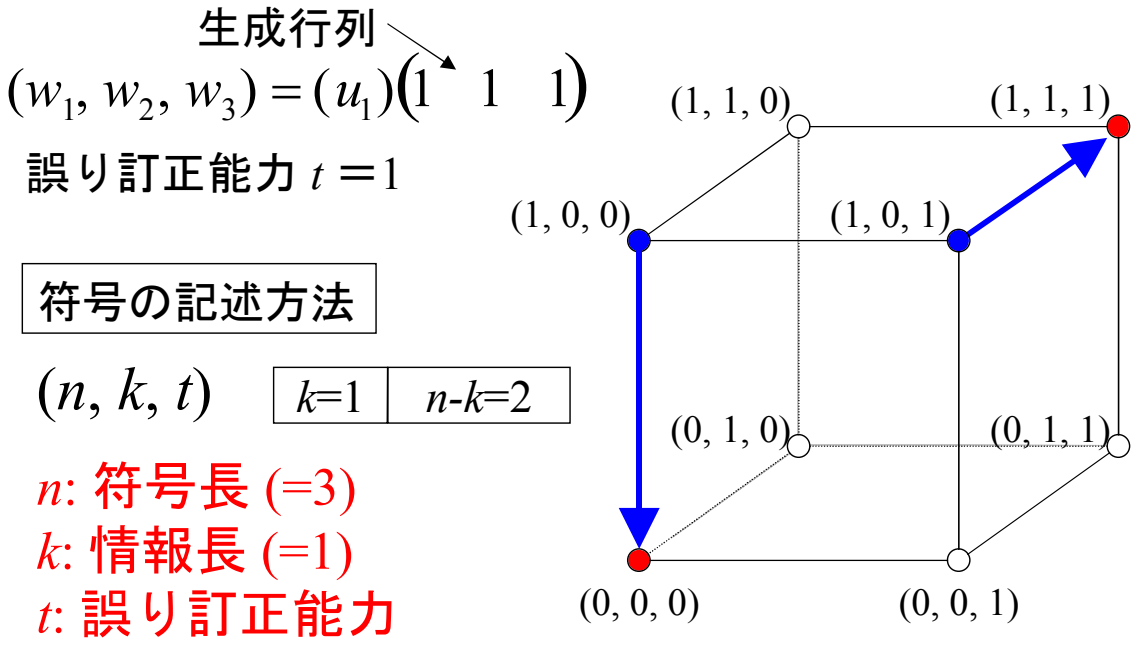


図 5-1-3-2　符号の誤り訂正能力

今回使用している符号は、誤り訂正符号ではなく、欠損を復元する符号である。符号の一部が欠損するということは、その符号アルファベットが、未知の値になることに相当する。２元符号の場合、「0」と「1」の中間の値である「1/2」と表現するのが(数値的に)妥当である。３次元空間で考えると、図 5-1-3-3 で示す様な座標となる。誤り訂正の場合と異なり、符号アルファベットのうち２個が欠損しても、符号語との距離から正しく復号することが可能となる。従って、最小 Hamming 距離と欠損復元能力との間には、誤り訂正の様に係数 1/2 がなく、最小 Hamming 距離から 1 を引いた値が欠損復元能力となる。

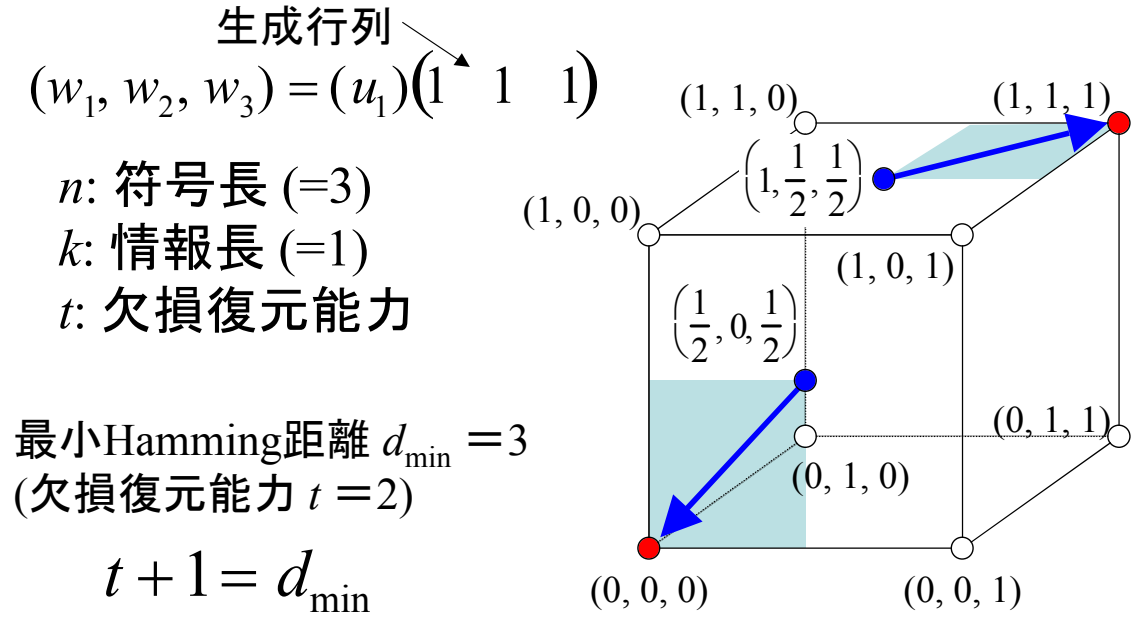


図 5-1-3-3　欠損復元符号の復元能力

本報告書では、(*n*, *k*, *t*)欠損復元符号と呼ぶ。欠損復元符号の能力を比較する場合、出来るだけ短い冗長部分(*n*-*k*)で、出来るだけ多くの欠損を復元することが望ましい。従って、評価パラメータとしては、*t*/(*n*-*k*)が出来るだけ大きい方が良いことが分かる。ただ、*t* は *n*-*k* を超えられないという Singleton 限界が存在するため、*t*/(*n*-*k*)の値は、1 に近いほど復元能力が高いことが分かる。しかし、単に *t*/(*n*-*k*)だけでは、議論できない。図 5-1-3-4 に示す様に、*n*=2, *k*=1, *t*=1 の場合と、*n*=200, *k*=100, *t*=100 の場合では、復元能力に大きな隔たりあることが分かる。つまり、*t*=100 の場合には、連続した 100 個の欠損にも耐えることが出来るが、*t*=1 の場合には、連続欠損に耐えることは出来ない。従って、この効果を考慮した比較が必要となる。一般に、この符号アルファベットを *n* 個集めたものを、符号化ブロックと呼ぶ。

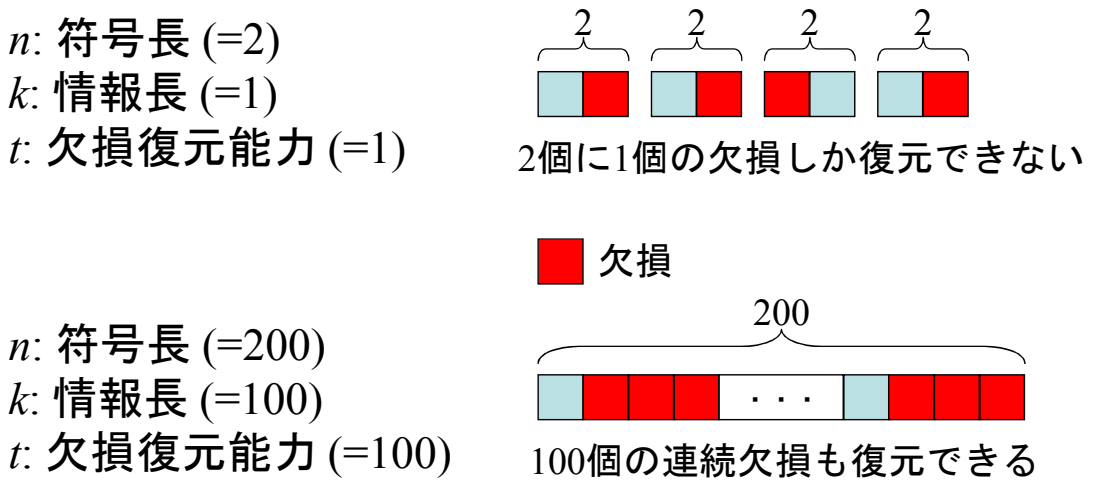


図 5-1-3-4　欠損復元能力の比較

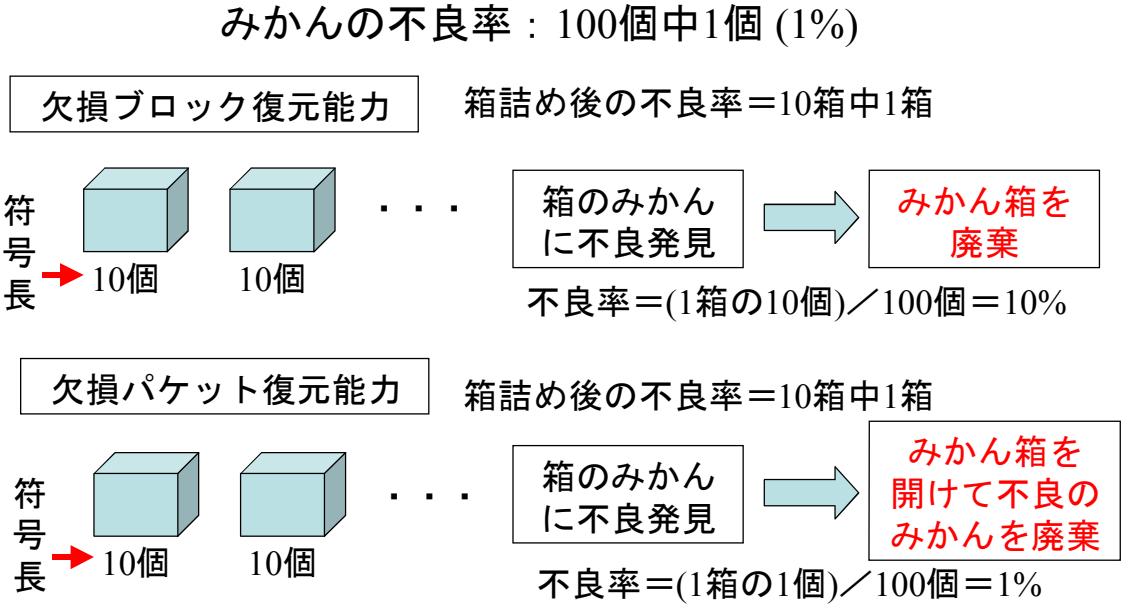


図 5-1-3-5　ブロック化の影響

符号化ブロックのサイズを大きくし、図 5-1-3-5 の様な影響が現れる。通常の欠損復元符号では、ブロックの復元に失敗した場合には、ブロックに含まれる本来欠損していないパケットまで、廃棄されてしまう。このため、ブロックサイズが大きくなるにつれて、ブロック復元率が低下してしまう。特に、パケット欠損率が大きい場合には、この影響が大きい。図 5-1-3-5 の例では、パケット欠損率が 10%になった場合には、ブロックとして想定されているミカン箱内に必ず不良(欠損)が発生するため、すべてのミカン箱全体が廃棄されてしまう。従って、ブロックの大きさを考慮した欠損復元符号の設計が必要となる。

5-1-3-2　ランダム送信方式

データ通信では、図 5-1-3-1(上)に示す様に、(当然のことながら)必要となるデータシンボルを本来の順番通りに相手側に送る(Sequential 送信方式)。本章で述べる「シンボル」とは、データを送り出す時の単位で、基本的には「パケット」と同様である。しかし今回の LT 符号等の基本的な考え方は、図 5-1-3-1(下)に示す様に、ランダムに選ばれたシンボルを順不同に送信するという方式を取っている(Random 送信方式)。あたかもサイコロを振って、シンボルを選んで送信するようなものである。Sequential 送信では、同じシンボルは(１周期中に)１回しか送出されないが、Random 送信では、確率的に同じシンボルを２回以上送信することもあり得る。

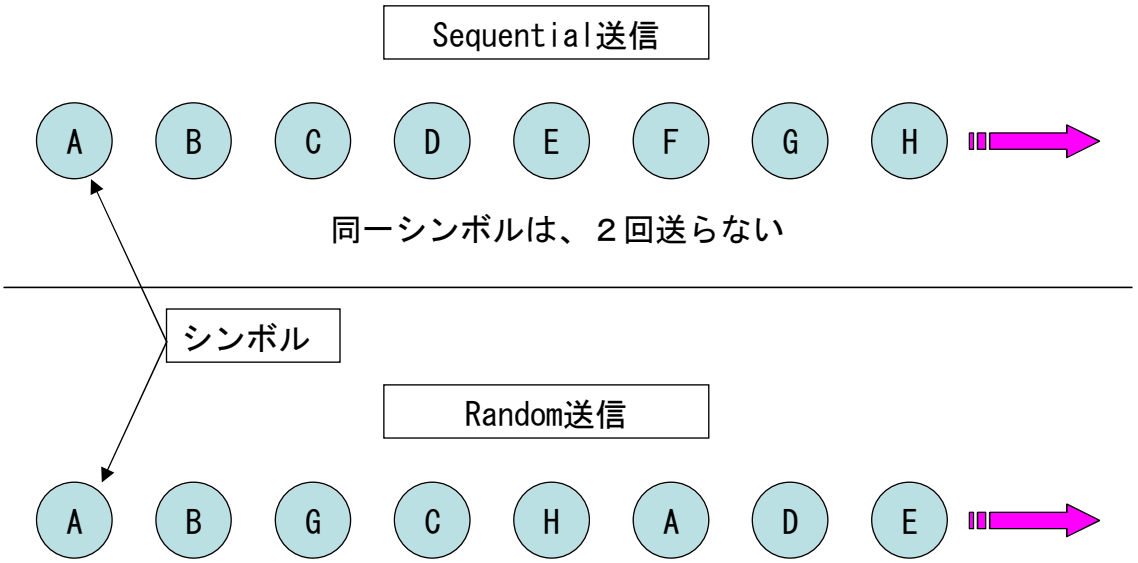


図 5-1-3-1　データの伝送方法

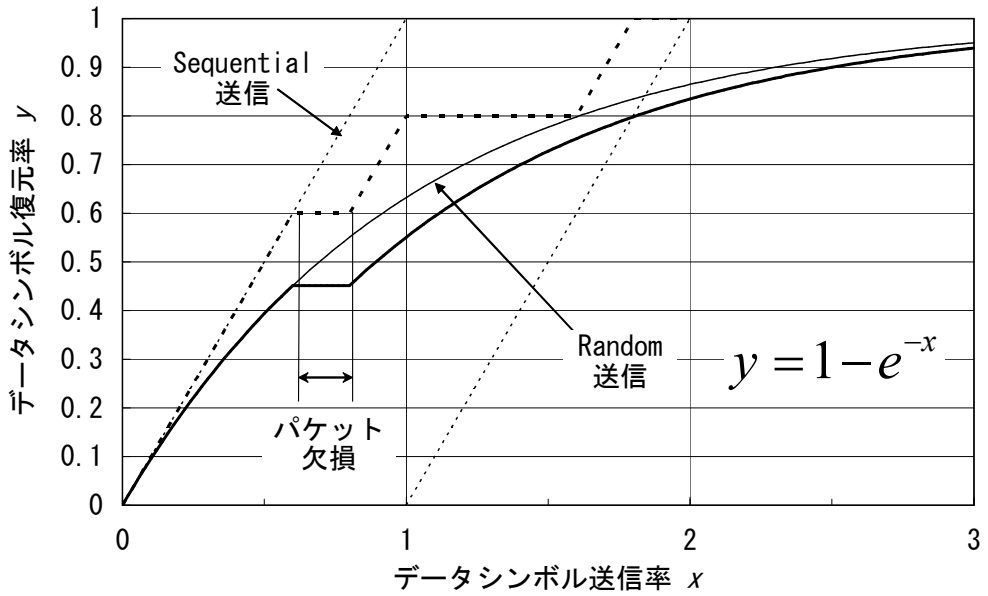


図 5-1-3-2　データシンボルの復元率

図5-1-3-2に示すようにSequential送信方式では、100%のデータシンボルを送信すれば、(伝送路にシンボルの欠損がない場合)受信側で100%のデータを復元できる。しかし、伝送路でシンボル(パケット)欠損が発生すると、その部分だけ、元のシンボルを復元(受信)できなくなる。TCP/IPを用いた伝送では、ARQ (Automatic Repeat reQuest)と呼ばれる自動反復要求(受信パケットに問題を検出した時に、再送を要求する通信方法)によって、欠損したシンボルが再送信される。この方法は、シンボル(パケット)に特別の工夫が不要で、パケットオーバヘ

ッドを回避できるという長所がある。しかし欠損が発生した場合、修復までデータシンボルが１往復する時間だけ、遅延が発生するという欠点を持っている。

受信側が、どのシンボルが欠損したかを送信側に伝えることが出来ない場合、すべてのデータシンボルを再送する必要がある。図5-1-3-2のSequential送信方式では、シンボル欠損が発生すると、全データの再送を開始し、欠損したシンボルが受信側に届いた時点で、伝送完了となる。従って、最初のシンボルが欠損した場合は、再送するデータが少なくてすむが、最後のシンボルが欠損した場合は、２倍のデータを送信する必要がある。一方Random送信では、シンボルの送信数xとデータの復元数yの間に、

$$y = 1 - e^{-x}$$

という関係がある。シンボル欠損が発生した場合、送信シンボル自体がランダムに送信されているので、途中で発生するランダムなシンボル欠損に対しても、この曲線の形状は変化しない。つまり、どのシンボルが欠損しても、このグラフが(欠損したシンボル数に応じて)右にシフトするだけであることが分かる。伝送路で発生するランダムなシンボル欠損によって、送信数-復元数の基本的な関係は変わらないというのが、Random 送信の大きな特徴である。

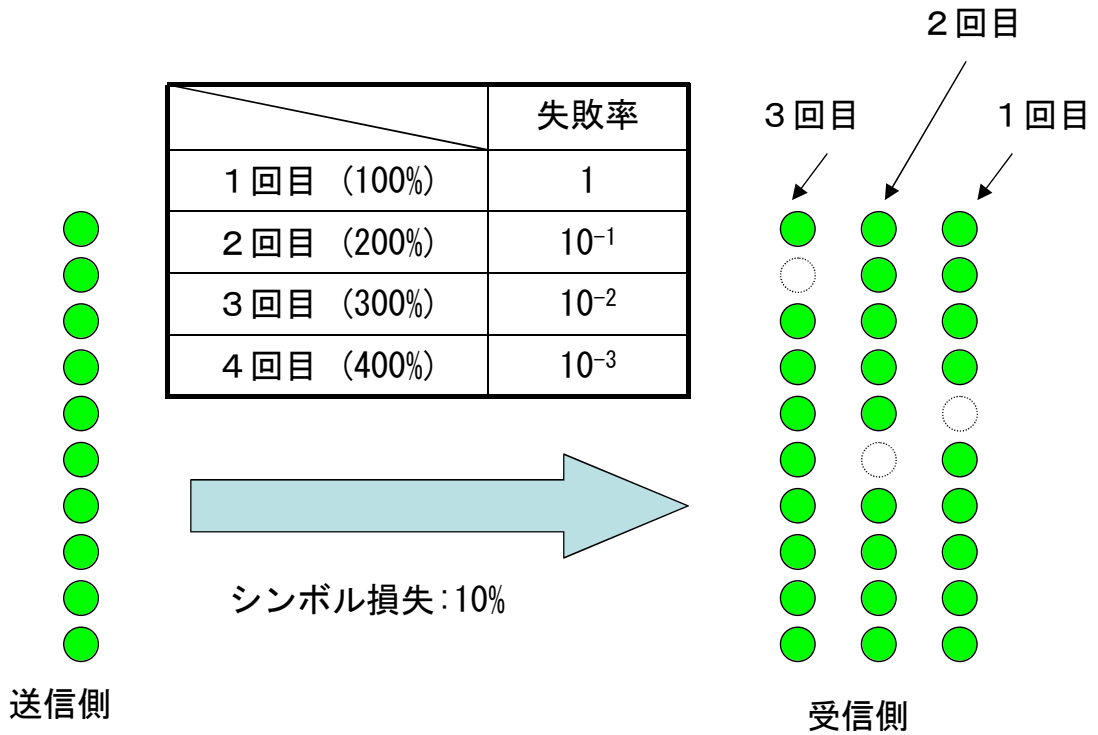


図5-1-3-3　もっとも単純なForward Error Correction

Sequential送信では、伝送途中で発生したシンボル欠損を補うために、データシンボルの送信が一巡した後、さらに同じデータシンボルを送信する。この方法による伝送失敗確率の計算例を、図5-1-3-3に示す。伝送路におけるシンボル欠損率が10%の場合、10個のシンボルを送信すると、平均的に9個のシンボルを受信することが出来る。平均的に見れば、必ず1個のシンボルが欠損するため、必ずデータ伝送に失敗(失敗率 : 100%)することになる。受信側から送信側へは、

ARQの様な情報のフィードバックがないため、送信側は受信側でどのシンボルが欠損したかを知ることは出来ない。このため、欠けたシンボルを復元するために、再度全シンボルを送信する。この際、前回の送信で欠損したシンボルが受信できれば伝送成功となるが、10%の確率で前回と同じシンボルが欠損するため、同じデータを一巡(送信)しても、受信側での成功率は100%とならない。つまり、200%のデータシンボルを送信しても、失敗率は10%(10^{-1})となる。

さらに図5-1-3-3に示すように、300%, 400%と送信シンボル数を増加させる毎に、失敗率は1/10に低下することが分かるが、送信シンボル数を増やしても失敗率は0%になることはない。ちなみにパケット(シンボル)伝送の失敗率が、このように0%にならないことは、FEC方式固有の現象ではない。ARQ方式においても、再送されるパケット(シンボル)自体にも欠損が発生する確率があるため、パケット(シンボル)欠損のある伝送路に対して失敗率は0%にならない。

5-1-3-3　Digital Fountain社誤り訂正符号の基本概念

今回我々が検討するDigital Fountain(DF)社の誤り訂正符号は、前節のような確率的な概念に基づいて作成されており、元のデータを確実に復元できるわけではなく、非常に低い確率ではあるが復元失敗率が存在する。つまり、何%かのシンボルを余分に送ることによって、受信側で(一定の失敗率以下で)欠損シンボルを復元する方法である。一方、Reed-Solomon等の誤り訂正符号は、訂正可能な誤り数以下であれば確実に誤りを訂正できる。

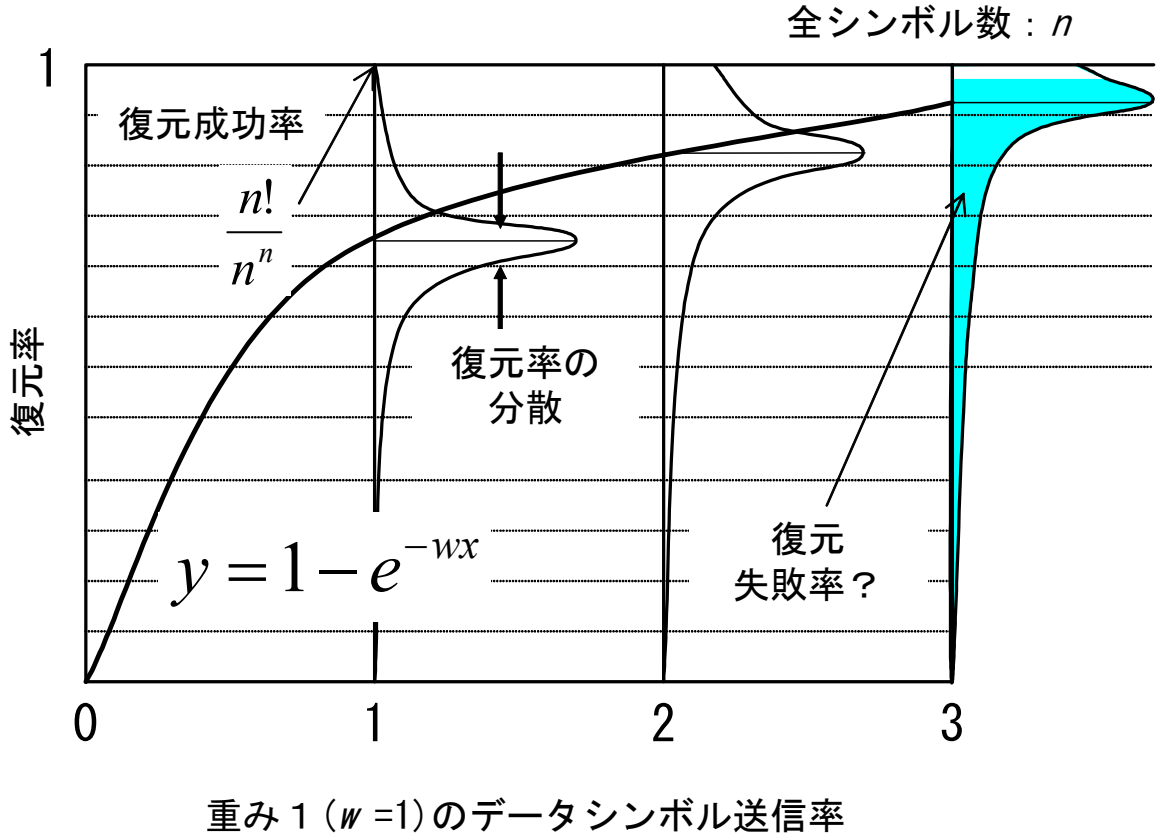


図5-1-3-4シンボル送信による復元率の分散

上記のような確率的な送信では、シンボル受信の成功率が100とはならないため、「無限の時間をかけてもすべてのデータを受信できない」という印象を受けることがある。シンボルが無限にある場合には、この考え方が成立するが、実際には有限個のシンボルを取り扱うため、復元率は図5-1-3-4に示すように、一定の分散をもつことになる。つまり100%のシンボルを受信した場合、1-1/*e*しかシンボルが復元できないような気がするが、実際には$n!/n^n$の確率で全てのシンボル送信に成功する。

DF社の誤り訂正符号では、基本的な概念は上記の通りであるが、実際には色々な工夫がなされている。元のシンボルをランダムに送信するだけでは、元のデータを復元するために送るデータシンボル数が、非常に多くなる。例えば、100%のデータシンボルを受信しても、平均的に63.2%の元データしか復元することは出来ない。

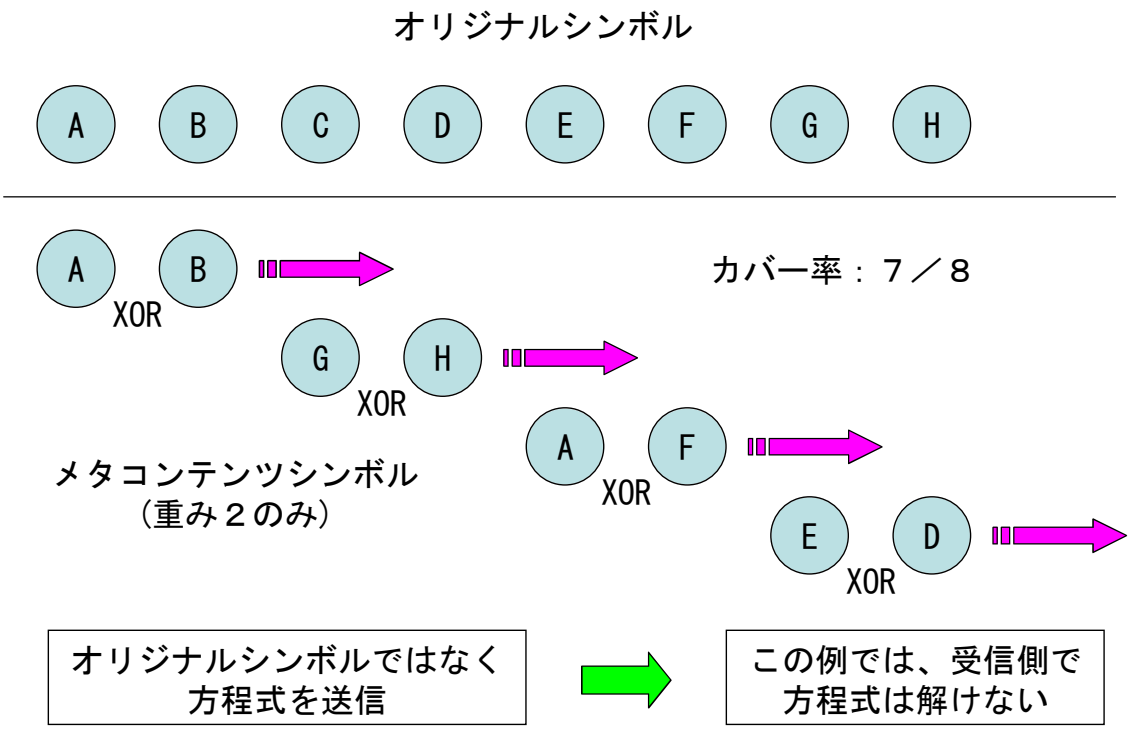
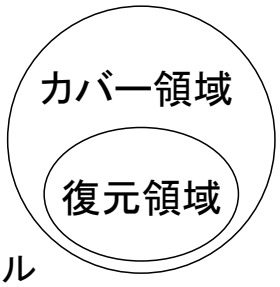


図5-1-3-5　メタコンテンツでの送信方法

そこで図5-1-3-5に示すように、2つのシンボルの排他的論理和(XOR)を取って、このシンボル(メタコンテンツシンボル)を相手側に送る。この場合、メタコンテンツの重みは2であり、何も演算しないで送信する場合は、重み1のメタコンテンツシンボルとなる。この様に、複数のコンテンツから生成されたメタコンテンツシンボルは、受信側で再度排他的論理和(XOR)演算を行うことによって、元のデータシンボルを復元できる。つまり方程式を順次送信し、それを受信側で解くことに相当する。従って、「受信した方程式の数」と「復元できたデータシンボル数」の関係は、状況によって異なり、一意に決めることは出来ない。この関係を明確にするために、「復元率」「カバー率」という2つの概念を導入する。図5-1-4-5の例では、全シンボル数が8個であり、C以外の7個のシンボルがカバーされているので、カバー率は7/8となる。



復元領域⊆カバー領域

復元率
メタコンテンツシンボルによって復元されたシンボル

カバー率
メタコンテンツシンボルで参照されているシンボル

メタコンテンツの重み	
全て１	全て２
復元率＝カバー率	復元率≠カバー率
メタコンテンツシンボルは全て復元	復元率＝０

図5-1-3-6　復元率とカバー率の関係

この定義では、復元率はカバー率より小さいか等しく、集合論的に言えば、カバー領域は復元領域を包含している。図5-1-3-6に示されるように、メタコンテンツの重みが全て1の場合は、復元率＝カバー率となるが、これ以外の場合は、両者の値は異なる。ちなみに、全てのメタコンテンツシンボルの重みが2の場合は、元のシンボルを復元することは出来ない(詳細は、後述の表5-1-3-3参照)。

図5-1-3-7は、メタコンテンツの(全ての)重みを変化させた場合の、メタコンテンツシンボル送信率xと受信シンボルカバー率yの関係である。平均重みをwとした場合には、

$$y = 1 - e^{-wx}$$

という関係が成立し、重みを増加させることによって、カバー率は急速に立ち上がることが分かる。つまりメタコンテンツの重みを増加させると、カバー領域は100%に近づき、元のシンボル復元の必要条件を満たしていることが分かる。

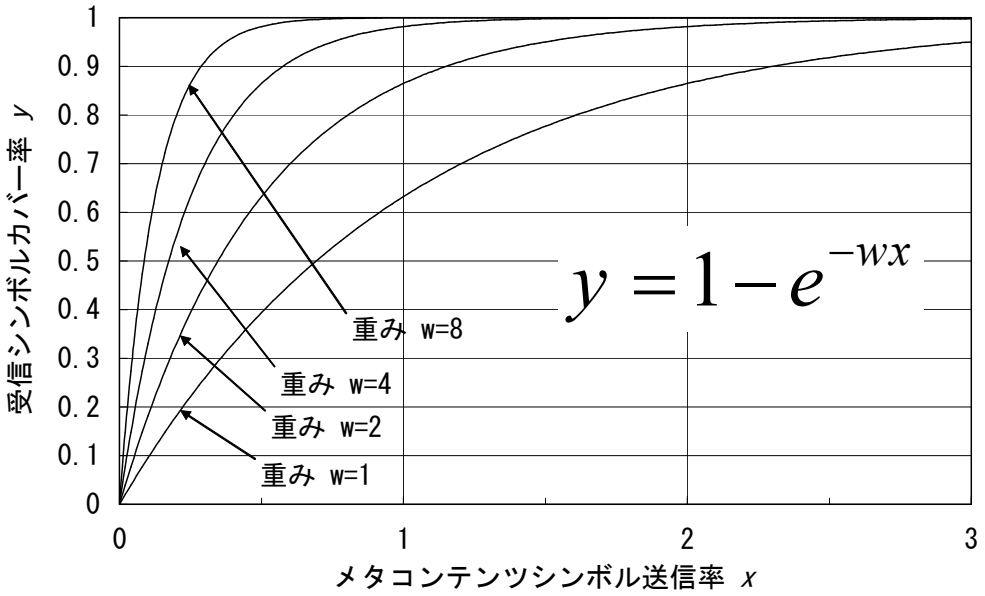


図5-1-3-7　メタコンテンツの重みを変化させた場合の復元率

図5-1-3-8は、集合論的に見た各領域の関係を示したものである。1から無限大までの重みを含んだメタコンテンツを送信した場合、受信側でのシンボル復元は、集合論的に4つの領域に分類される。

（１）重み1のメタコンテンツでのみカバーされる領域(復元可)

（２）重み1と重み2以上のメタコンテンツ両方でカバーされる領域(復元可)

（３）重み2以上のメタコンテンツのみでカバーされる領域(復元？)

（４）どのメタコンテンツからもカバーされない領域(復元不可)

(２)と(３)の領域をまたがって参照しているメタコンテンツシンボルがあれば、そのメタコンテンツによって、(３)の領域での元のシンボルを復元することが可能となる。この様な復元方式は、連鎖反応コード(Chain Reaction Code)と呼ばれている。

カバー関数

$$y = 1 - e^{-wx}$$

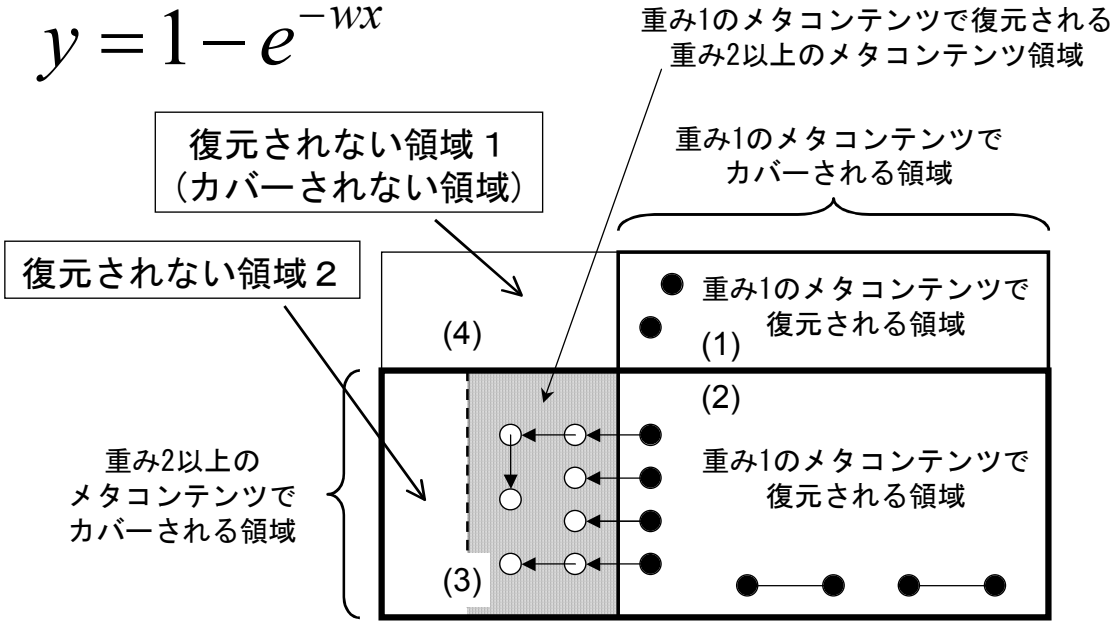


図5-1-3-8　集合論的に見た各領域の関係

また図5-1-3-8に示すように、元のシンボルが復元されない領域は、2つに分類される。

＜復元されない領域１＞　　上記(４)に対応する領域

復元されない領域１の減少方法としては、「重み1のメタコンテンツを増加させる」方法と「重み2以上のメタコンテンツを増加させる」方法が考えられるが、図5-1-3-7に示すように関数値の増加が早い「重み2以上のメタコンテンツを増加させる」方法が有利であると考えられる。

＜復元されない領域２＞　　上記(３)のうち連鎖反応で復元されない領域

復元されない領域2の減少方法としては、シンボルによる連鎖反応が促進するように、重みの分布(重み関数)を最適化する必要がある。つまり、重み1のメタコンテンツシンボルを受信した場合に、復元できる元のシンボル数が多いほど良いことが分かる。

5-1-3-4　Digital Fountain社誤り訂正符号の符号化／復号化

図5-1-3-9に、Digital Fountain(DF)社で開発されたLT codingのアルゴリズム(送信側)を示す。この例では、入力データである元のシンボル数は、a～hの8個で、一定数の列を発生するマトリックスによって選ばれたシンボル同士で排他的論理和を取り、これをメタコンテンツとして送信する。平均的な重みは、受信側での復元率が高まるように設定する必要がある。またこの演算の関係は、図5-1-3-9(右)に示すように、グラフ表示することも可能である。入力シンボルは白丸で表され、出力シンボルは黒丸で表される。例えば、a xor gという出力シンボルは、入力シンボルaとgとに繋がっており、この両者の排他的論理和で構成されていることを示している。また、この接続線の数は、「排他的論理和演算を行

った回数＋受信メタコンテンツ数」を示しており、符号化／復号化に必要な処理時間と関係していることが分かる。

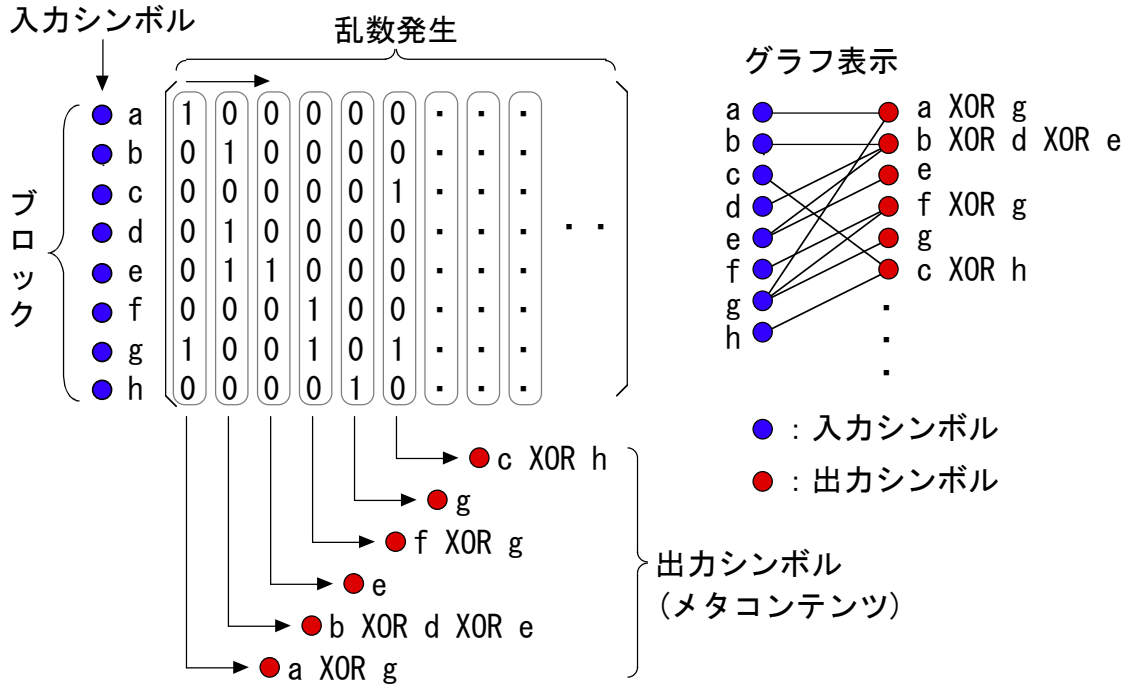


図5-1-3-9　LT codingの符号化アルゴリズム

図5-1-3-10は、受信側での復号アルゴリズムである。図5-1-3-9で生成されたメタコンテンツシンボルのうち、最初の2つを受信した状態では、どの入力シンボルも復元することは出来ない[ステップ１]。ここで、重み1のシンボルであるeを受信すると、元のシンボルeを復元することが出来ると同時に、eというシンボルを含んだ他のメタコンテンツシンボルの重みを1つ下げることが出来る。この例では、b xor d xor eというメタコンテンツを、b xor d xor e xor e=b xor dという演算によって、b xor dにすることが出来る[ステップ２]。また排他的論理和演算によって、接続線の数は減少する。さらに、gという重み１のメタコンテンツを受信することによって、gを参照しているa xor g, f xor gという２つのメタコンテンツから、aならびにfを復元することが出来る[ステップ４,５]。この処理を続けることによって、最終的に全てのシンボルを復元することが可能となる。ただこれは確率的な処理であり、何個のメタコンテンツを受信する必要があるかは、一意には決まらない。メタコンテンツは連立方程式であるため、少なくとも元のシンボル数以上のメタコンテンツシンボルを受信する必要があるが、その中には同じメタコンテンツシンボルを2回以上重複して受信している場合があり、必要数は確率的に決定される。

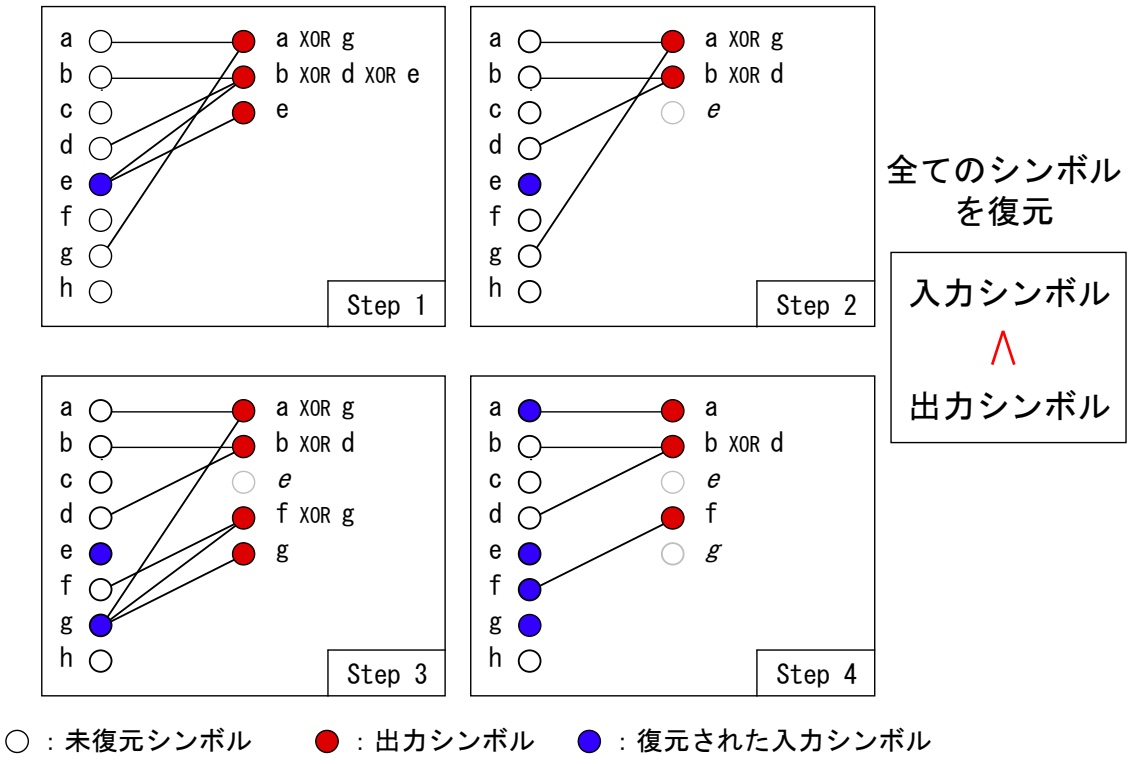


図5-1-3-10　LT codingの復号化アルゴリズム

表 5-1-3-2 に示すように、メタコンテンツの重複受信は「明示型」と「暗黙型」の 2 つに分類できる。明示型の場合は、明らかに同じメタコンテンツを受信する場合であり、重みが大きくなるほど受信確率は小さくなる。一方暗黙型は、A xor B と B xor C を受信している状態で、C xor A というメタコンテンツを受信しても、以前の 2 つのメタコンテンツから導出できるので、意味のないメタコンテンツとなる。受信したメタコンテンツが参照したシンボルを 1 で示し、参照されないシンボルを 0 で示すマトリックス(図 5-1-3-9)で表現すると、その階数(rank)が元のシンボル数に等しくなった時、元のシンボルを復元できることになる。

表 5-1-3-2　メタコンテンツの重複受信

MC シンボル 重複受信の型	内　容
明示型 (Explicit)	同一MCシンボルを、２回受信 A 受信→再度A 受信 A xor B xor C 受信→再度 A xor B xor C 受信
暗黙型 (Implicit)	同一MCシンボルではないが、受信側にとって 意味のないMCシンボルを受信 A xor B, B xor C 受信→A xor C 受信 A xor B xor C, B xor C 受信→A 受信

MC：メタコンテンツ

表 5-1-3-3 は、復元のためのアルゴリズムを示している。シンボルの重みが全て 1 の場合、復元操作を行うことなく元のシンボルを復元できる。すべてのメタコンテンツの重みが 2 以上の偶数の場合、メタコンテンツ同士の演算を繰り返しても、最低の重みは 2 以上となり、元のシンボルを復元することは出来ない。重みが 3 以上の奇数の場合、メタコンテンツによっては、元のシンボルを復元することが可能となる。重みが 1 以上のメタコンテンツシンボルが混じった状態で受信した場合は、3 つのアルゴリズムが考えられる。

(１)重み 1 のメタコンテンツ受信を待って、復元のトリガとする

重み 1 のメタコンテンツを受信することによって、連鎖反応が進み、元のシンボルを復元する。

(２)復元シンボルと受信メタコンテンツシンボルを比較し、重みを減少させる

内部で復元されたシンボルと受信したメタコンテンツを比較し、受信したメタコンテンツの重みを減少させ、重み 1 になった場合は、復元処理を開始する。

(３)メタコンテンツ同士を比較し、重みを減少させる

他のメタコンテンツコンテンツシンボルを包含しているようなメタコンテンツシンボルを検索し、相互に重みを減少させる。

表 5-1-3-3　メタコンテンツ復元のためのアルゴリズム

<table>
<tr><td>シンボルの重み</td><td colspan="2">復元方法</td></tr>
<tr><td>全て１</td><td colspan="2">復元不要 (受信と同時に復元)</td></tr>
<tr><td>全て２</td><td colspan="2">復元不可能 (重み１生成不可)
A xor B, B xor C→A xor C</td></tr>
<tr><td>全て奇数
(３以上)</td><td colspan="2">A xor B xor C, B xor C xor D, A xor C xor D
→A xor D, A xor C xor D→C</td></tr>
<tr><td>全て偶数
(４以上)</td><td colspan="2">復元不可能 (重み１生成不可)
A xor B xor C xor D, B xor C xor D xor E
→A xor E</td></tr>
<tr><td rowspan="3">重み１-L
(1,2,3,4・・)</td><td>受信</td><td>重み１シンボルの受信を待つ</td></tr>
<tr><td rowspan="2">内部
生成</td><td>シンボルとMCシンボルを比較して重みを減少
B, C：既知，A xor B xor C 受信</td></tr>
<tr><td>MC シンボル同士を比較して重みを減少</td></tr>
</table>

MC：メタコンテンツ

表 5-1-3-4 は、シンボルが「A」「B」「C」の３つで、重みがすべて２のシンボルを送った場合の、メタコンテンツデータの組合せである。簡単のために、シンボル長は 1bit としてある。この図に示されるように、メタコンテンツのデータは縮退し、このメタコンテンツシンボル情報のみでは元のコンテンツを復元できないことが分かる(例：(A, B, C)=(0, 0, 0)と(1, 1, 1)を区別できない)。

表 5-1-3-4　メタコンテンツの内容(シンボル数：３，重み：すべて２)

A	B	C	A xor B	B xor C	C xor A
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	1	1	0
0	1	1	1	0	1
1	0	0	1	0	1
1	0	1	1	1	0
1	1	0	0	1	1
1	1	1	0	0	0

5-1-3-5　重み分布によるメタコンテンツ受信数

メタコンテンツの重みは、一定の値ではなく、重み１からある有限個数まで

を一定の比率で配分することが有効である。今回、特定の重み分布に対して、どの程度余計にメタコンテンツシンボルを受信すればよいかを、シミュレーションを行った。

表 5-1-3-5 は、重み関数の分布を示したものである。基本的な関数として、

$$\frac{1}{n-1}-\frac{1}{n}$$

を用いている。この関数は、2 から無限大までの総和は、1 となる。

$$\sum_{i=2}^{\infty}\left(\frac{1}{i-1}-\frac{1}{i}\right)=1$$

今回のシミュレーションに用いた重み関数は、係数 L, r, q を変化させることによって、重み関数の分布を変更することが出来る。

表 5-1-3-5　重みの確率分布

重み	発生確率
1	[1-r+{r/(L-1)}]*q
2	{(1/1)-(1/2)}*r
3	{(1/2)-(1/3)}*r
4	{(1/3)-(1/4)}*r
・ ・	・ ・
L-3	{1/(L-4)-1/(L-3)}*r
L-2	{1/(L-3)-1/(L-2)}*r
L-1	{1/(L-2)-1/(L-1)}*r
L	[1-r+{r/(L-1)}]*(1-q)

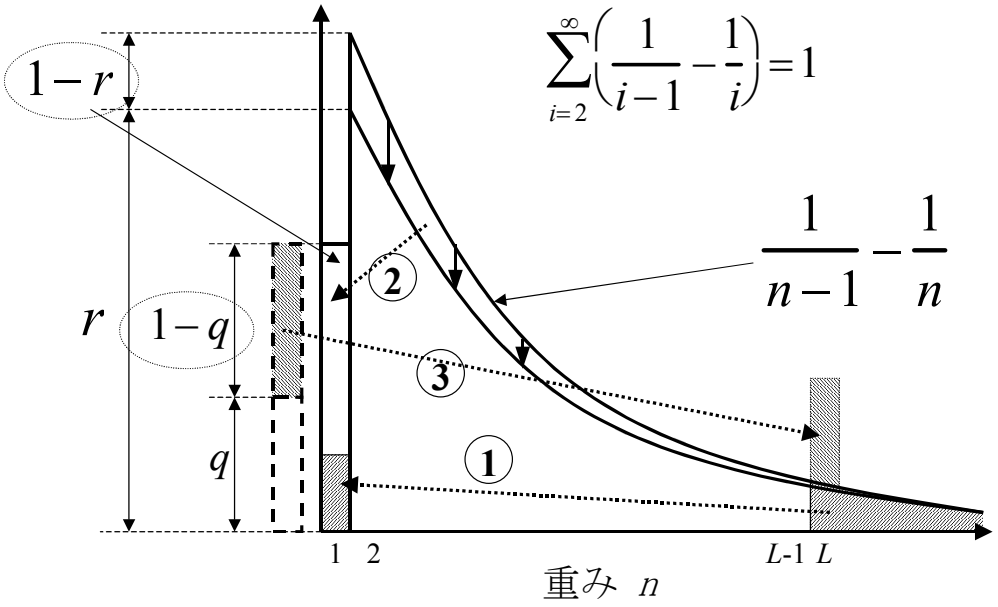


図 5-1-3-11　重み関数の意味

図 5-1-3-11 は、表 5-1-3-5 に示された重み関数の意味を示したものである。重み関数は、無限大までの重みを取るのではなく、一定の値(L-1)で打ち切り、打ち切られた部分を、重み 1 の発生確率に割り当てる。さらに、重み 1 の発生確率を変化させるために､重み関数の 2 から L-1 の部分に係数 r をかけて減少させ、余った部分を重み 1 に割り当てる。さらに、最大重み L の発生確率を変化させるために、重み 1 の発生確率を q : (1-q)の比率で最大重みに割り当てる。この関数を用いて、メタコンテンツを復元するのに必要なメタコンテンツシンボル数をシミュレーションにより算出した。

表 5-1-3-6　カバー率と重みの関係 (r=0.9, L=255)
(元のシンボルと同数のメタコンテンツシンボルを受信した場合)

q	重み 2 以上で カバーされない領域	平均重み	重み 1 の割合	重み 255 の割合
1.0	$4.081*10^{-3}$	5.60	10.4%	0%
0.9	$2.911*10^{-4}$	8.23	9.3%	1.0%
0.7	$1.482*10^{-6}$	13.50	7.2%	3.1%
0.5	$7.540*10^{-9}$	18.75	5.2%	5.2%
0.3	$3.837*10^{-11}$	24.01	3.1%	7.2%
0.1	$1.954*10^{-13}$	29.27	1.0%	9.3%
0.0	$1.388*10^{-14}$	31.90	0%	10.4%

シミュレーションを実施する前に、パラメータ q を変化させて、図 5-1-3-8 の「復元されない領域 1」がどの程度になるかを計算した。パラメータ q を変化させることによって、重み 1 と最大重みの割合ならびに平均重みを求め、カバー

されない領域を算出したのが、表 5-1-3-6 である。平均重みを増加させると、カバーされない領域が低下することが分かる。図 5-1-3-12 は、平均重みとカバーされない確率との関係を示したものである。縦軸を対数表示すると、両者の関係は、直線で表されることが分かる。

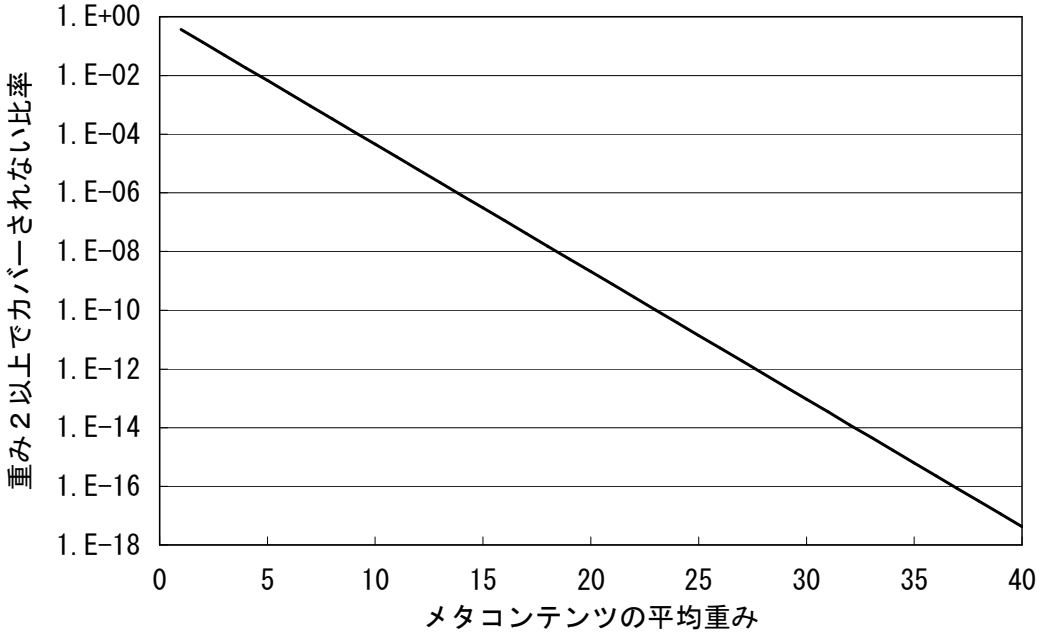


図 5-1-3-12　平均重みとカバーされない比率との関係

5-1-3-6　シミュレーションの結果

図 5-1-3-13 は、シミュレーションにおいて生成されたメタコンテンツシンボルと、復元されたメタコンテンツシンボルの関係を示したものである。元のシンボル数は 65,536 個で、最大重みは 255 である。シミュレーションの回数は、それぞれ 1 回であるが、パラメータの絞り込みを厳密に行うためには、数百回以上の試行が必要だと思われる。

図 5-1-3-13 でプロットが途切れているところで、元のシンボルが復元できたことが分かる。図 5-1-3-13 では、メタコンテンツシンボル数を元のシンボル数より 1.2〜1.8 倍程度生成しないと､元のシンボルを復元できないことが分かる。この様に、元のシンボルを復元するために余分に必要なメタコンテンツシンボル数を、オーバヘッド・メタコンテンツシンボル数と呼ぶ。表 5-1-3-5 に示されるように、q の値を小さくすると、重み 1 の発生割合が小さくなる。図 5-1-3-13 では、q の値を小さくすると、65,536 個(100%)のシンボルを受信した場合の復元率が小さくなっていることが分かる。しかし、65,536 個(100%)のシンボルを受信した後、急激に復元率が高まり、元のシンボルを完全に復元するのに必要なオーバヘッド・メタコンテンツシンボル数が減少していることが分かる。つまり、重み 1 のメタコンテンツを減少させた方が、少ないオーバヘッドで、元のシンボルを復元できることが分かる。ちなみに、r=0.89, q=0.1, L=255 の場合、重み 1 のメタコンテンツシンボル発生割合は、表 5-1-3-5 より 1.14%となる。

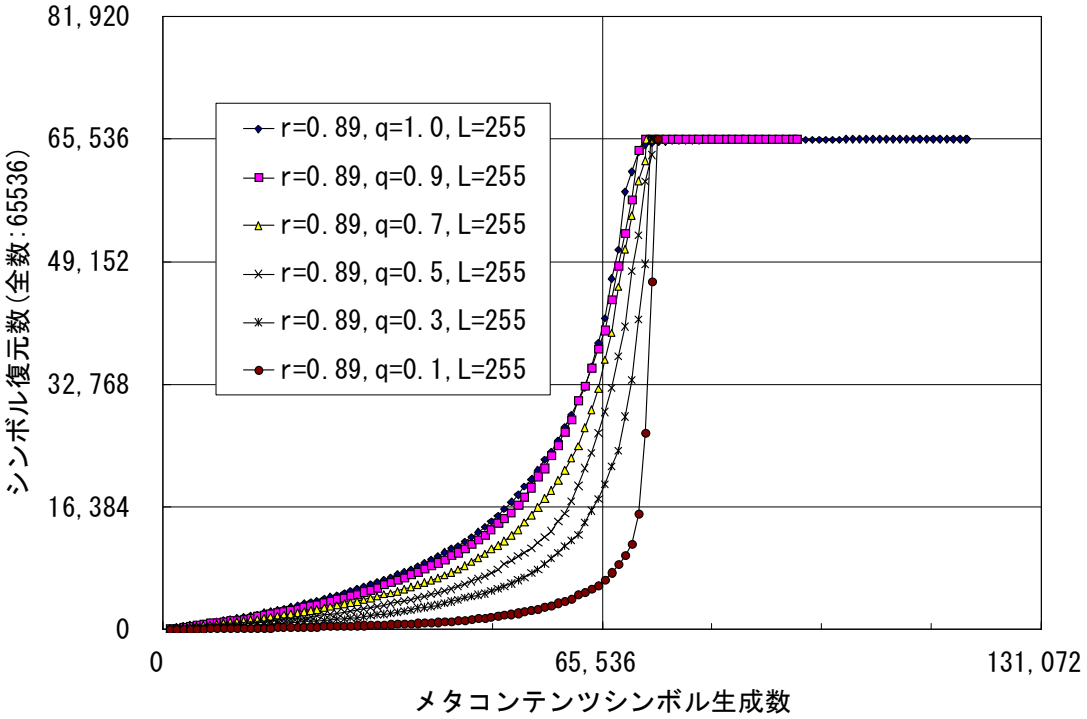


図 5-1-3-13　生成シンボル数と復元シンボル数（１）

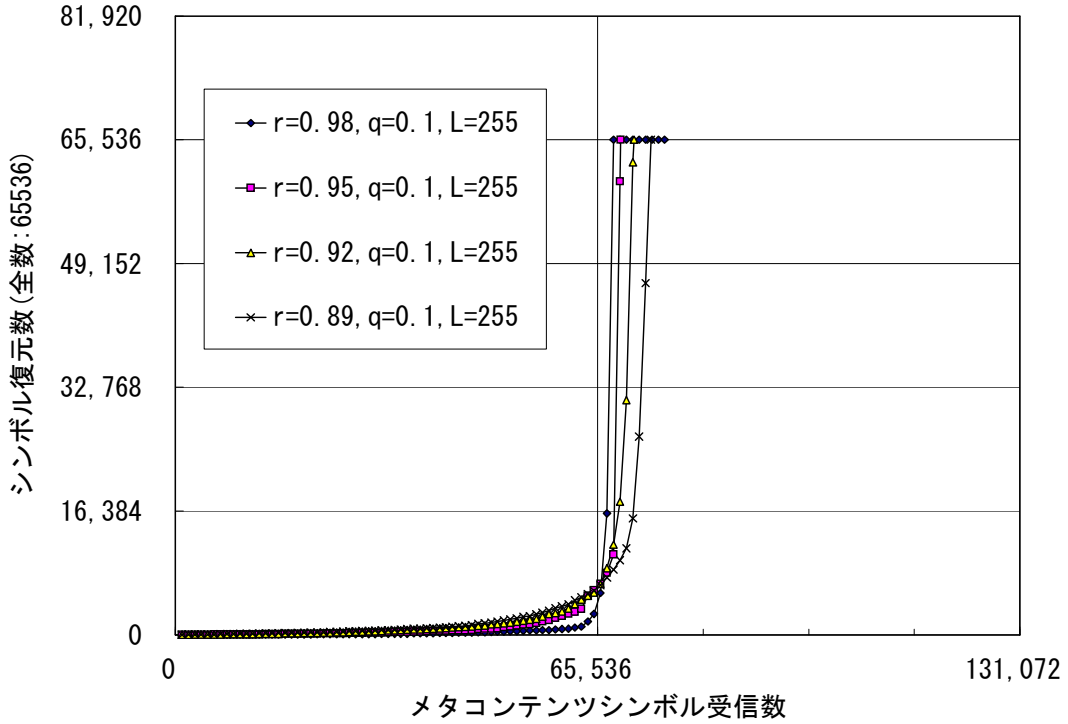


図 5-1-3-14　生成シンボル数と復元シンボル数（２）

次に q=0.1 とし r を変化させた時、復元に必要なオーバヘッド・メタコンテンツシンボル数をシミュレーションで求めたものが、図 5-1-3-14 である。この場

合、r=0.92 の時に、オーバヘッド・メタコンテンツシンボル数が最小になった。ちなみに、r=0.92, q=0.1, L=255 の場合、重み 1 のメタコンテンツシンボル発生割合は、表 5-1-3-5 より 0.84%となる。

図 5-1-3-15 は、これらのシミュレーション結果をまとめたものである。横軸は、q の値によって決まるシンボルの平均重みで、左側の縦軸は、元のシンボル数に対するオーバヘッド・メタコンテンツシンボル数の割合を示したものであり、右側の縦軸は、復号にかかった時間(Dec 処理時間)を示したものである。当初予想されたとおり、シンボルの平均重みを増大させていくと、オーバヘッドが減少していき、逆に Dec 処理時間は増加傾向にあることが分かる。ただ、Dec 処理時間曲線で、平均重みが最大の場合に、Dec 処理時間が下がっているが、この原因については現在のところ不明である。

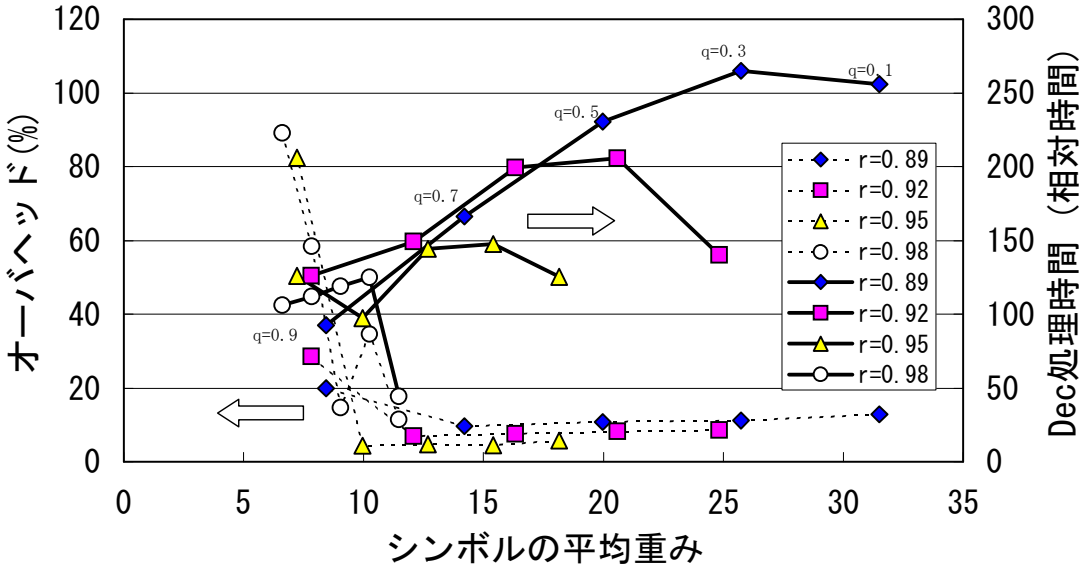


図 5-1-3-15　シンボルの平均重みと
オーバヘッドならびにデコーダ処理時間の関係

5-1-4　誤り訂正符号の性能比較

今回我々が取り上げた Digital Fountain 社の誤り訂正符号(LT 符号と、その改良型である Raptor 符号)と、一般に広く用いられている Reed-Solomon 符号の性能について比較を行う。それぞれの符号は、符号ごとに異なる特徴を持ち、用途に対して性能が異なるため、直接的に比較することは難しい。今回は、送出されたパケットが欠損した場合の修復能力という点で、Reed-Solomon 符号と Raptor 符号を比較する。

5-1-4-1　Reed-Solomon 符号と Raptor 符号

表 5-1-4-1 は、Raptor 符号と Reed-Solomon 符号を比較した結果を、簡単にまとめたものである。Reed-Solomon 符号は、連続したビット誤り(つまりバイト誤り)を訂正するように設計された符号であり、消失したパケットを復元する機能

はない。また符号化／復号化に際しては、有限体の中で多項式演算を行うことによって、誤り訂正に必要な冗長部分を計算するため、必要な処理時間が情報長の自乗に比例するという特徴を持っている。情報長が 200Byte 程度の場合、Raptor 符号と同一のハードウェアを用いて、Reed-Solomon 符号をソフトウェア的に処理した場合、１００倍以上の処理時間が必要になる。この様に大きな処理時間がかかると、符号化／復号化の処理が終了しないうちに次のデータが到着するという問題が発生する。従って、処理速度が情報長に比例する([O(n)]である)Raptor 符号が、処理能力において優れていることが分かる。

表 5-1-4-1 誤り訂正符号の比較

	Raptor符号	Reed-Solomon符号
符号化による効果	欠損パケットを復元	バイト誤りを訂正
符号化／復号化	ソフトウェア O(n)	ハードウェア O(n^2)
設定変更	容易	困難

今回のプロジェクトでは、ネットワークの誤り率に応じて自律的に誤り訂正能力を変更させることの出来る誤り訂正符号が必要となる。Reed-Solomon 符号では、複雑な多項式演算や復号アルゴリズムによって、専用の ASIC や FPGA (Field Programmable Gate Array)で処理されるために、Reed-Solomon 符号自体の誤り訂正能力を変更することが出来ない。ただ専用のハードウェア(ASIC or FPGA)の外部に設置されたソフトウェア的な処理によって、誤り訂正能力を変更させることは可能であり、以下のような方法が考えられる。

(1) Reed-Solomon 符号＋インターリーブ法

(2) Reed-Solomon 符号＋パンクチャド法＋インターリーブ法

インターリーブ法は、Reed-Solomon 符号を基にインターリーブを行ってブロック化し、誤り訂正能力を改善させる方法であり、バイト単位の誤り訂正能力を欠損パケットの復元能力に高めることが出来る。パンクチャド法は、Reed-Solomon 符号の特定のバイトを取り除き(パンクチャドし)、その符号をベースにインターリーブを行う方法である。この方法では、どのバイトを取り除くかによって、誤り訂正個数が大きく低下するので、注意が必要である。誤り訂正個数が低下した場合でも、符号長も低下するので、全体としては誤り訂正能力が向上する場合がある。

5-1-4-2　パケット欠損の発生原因

インターネットで発生するパケット欠損は、以下のように２つに分類される。

(1) 伝送路でのノイズによる(パケット)損傷によるパケット廃棄

(2) 中継器機(ルータ等)の輻輳によるパケット廃棄

ノイズによるビット誤り(パケットの損傷)は、0,1 レベルの分布が正規分布(ガ

ウス分布)であると仮定し、しきい値のレベルを超えた場合に誤りが発生すると考えるとガウス分布の積分となり、補誤差関数になることが分かる(図 5-1-4-1 参照)。

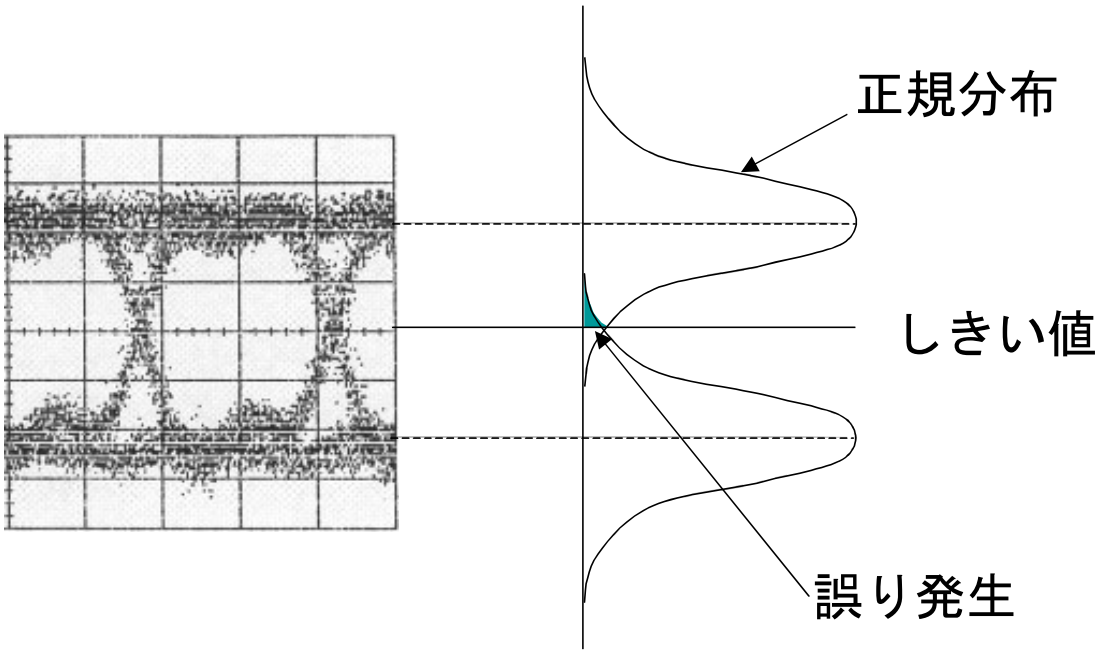


図 5-1-4-1　ノイズによる誤り発生のメカニズム

$$\mathrm{erfc}(x)=\frac{2}{\sqrt{\pi}}\int_{x}^{\infty}e^{-t^{2}}dt$$

一般的には、上記で示された補誤差関数で表される曲線が、誤り訂正を行うことによって、(同等の誤り率を実現する場合に必要な)SN 比が何 dB 低下したかを調べ、その値を符号化利得と定義し、符号の良し悪しを判断している。しかし、中継器機(ルータ等)の輻輳によるパケット廃棄を考えた場合、SN 比という概念は適切ではない。

今回我々は図 5-1-4-2 に示すように、符号化利得ではなく、誤り訂正の前後で誤り率がどの程度低下するかという「誤り訂正能力」を用いて、符号の良し悪しを検討する。厳密には、パケットをブロック化するので、横軸に平均的なパケット欠損率を取り、縦軸にブロック復元失敗率を取る。この場合、「誤り訂正能力」ではなく「ブロック復元能力」を比較することになる。

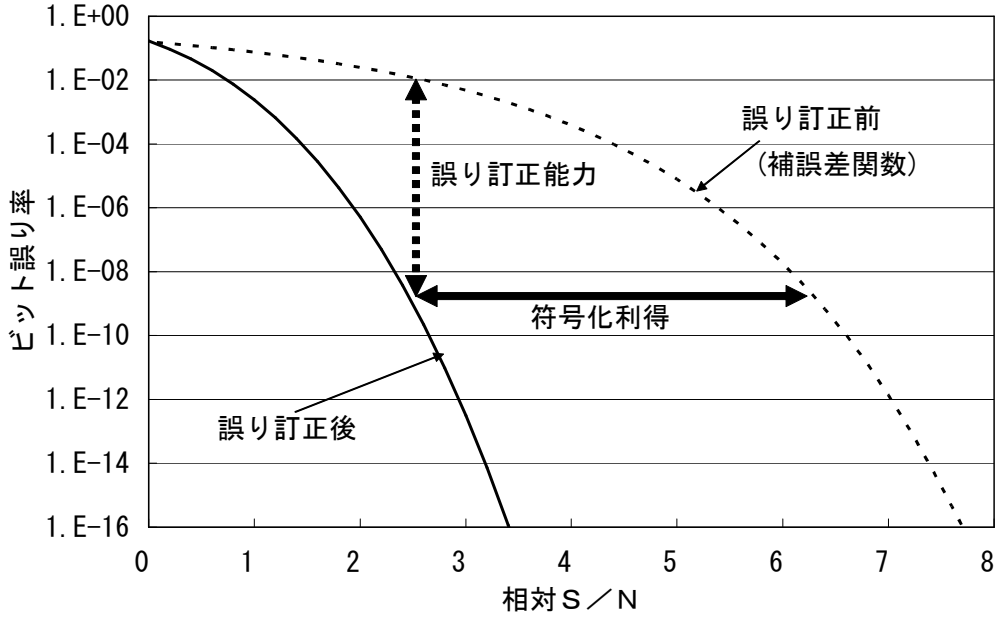


図 5-1-4-2　符号化利得と誤り訂正能力の関係

インターネットでの平均的なパケット欠損率を仮定し、複数のパケットをまとめてブロックとして取り扱った場合、その中に発生するパケット欠損は、一定の幅を持った二項分布 $f(p, n, t)$ となることが知られている。

$$f(p,n,t) = \sum_{i=t+1}^{n} {}_nC_i \cdot p^i (1-p)^{n-i}$$

p：平均的なパケット欠損率
n：ブロック内のパケット数
t：ブロック内での復元可能なパケット数

（１）受信側で誤り訂正を行わなかった場合

受信側で誤り訂正を行わなかった場合、上記のブロック内でパケット欠損が発生した場合、そのブロックの受信に失敗したと考え、そのブロック全体を廃棄する。ブロックの中には、受信に成功したパケットも含まれているが、このように定義すると、「ブロック復元失敗率」と「パケット欠損率」とが等しくなり、計算が簡単になる。

（２）受信側で誤り訂正を行う場合

ブロック内のパケット欠損数が設定された値以下であれば、ブロック内のパケットを復元できる。設定値を超えるパケット欠損がブロック内で発生した場合には、ブロックの復元に失敗したと判断する。

5-1-4-3　ブロック復元能力

平均的なパケット欠損率を横軸に取り、ブロック化を行った場合のブロック復元失敗率を縦軸に取って比較することにより、誤り訂正(欠損復元)符号の能力を把握することが出来る。誤り訂正を行わない場合は、パケットの(平均)欠損率とブロックの復元失敗率の関係は、傾き 1 の直線になるような気がするが、実際には、ブロック内に含まれるパケット数 *Ns* (統計でいうサンプル数)により、復元失敗率が増加することが分かる。パケットの(平均)欠損率(統計学でいう母集団の欠損率)を f_p とすると、誤り訂正を行わない場合のブロック復元失敗率 f_B は

$$f_B = 1-(1-f_p)^{N_s}$$

となり、f_p が小さい時には、

$$f_B \cong N_s f_p \quad (f_p << 1)$$

f_p が 1 に近づくと、

$$f_B \to 1 \quad (f_p \to 1)$$

となることが分かる。受信側で誤り訂正を行うと、ブロック復元失敗率 f_B は低下し、この結果から符号の誤り訂正能力を求める。

Raptor 符号では、入力データをシンボルという単位に分割し、そのシンボル同士の排他的論理和(XOR)を取ることによって、出力シンボルを作成し、それをパケットとしてまとめて送出している。さらにこのパケットをブロック化することによって、パケット欠損に対応している。特別な場合は、1 シンボルを 1 パケットで送出し、この場合シンボルとパケットは、同等のものと考えることが出来る。受信側で必要な出力シンボル数は、入力シンボル数以上必要となり、[余分に必要なシンボル数]／[入力シンボル数]をオーバヘッドと呼ぶ。今回の計算では、オーバヘッドが 5%の場合、10^{-13} の失敗率で元の入力シンボルを復元できると仮定して計算を進める。

図 5-1-5-3 に、Raptor 符号を用いた場合の伝送系を模式的に示している。まず設計パラメータとして、伝送路での平均的なパケット欠損率を設定する。さらにブロックを構成するパケット数を決めると、ブロック内で発生する欠損パケット数の分布が決まる。この分布は、二項分布であることが知られている。ブロックサイズを大きくすると、ブロック内での欠損パケット数の分布は、平均的なパケットの欠損率に収束する。つまり欠損パケット数の分布は、インパルス関数となる。ブロック内のパケット数を少なくすると、二項分布が広がり、伝送路に設定された(平均パケット欠損率)以上のパケット欠損率が発生する割合が増加し、欠損パケットを復元できない確率が増大する。

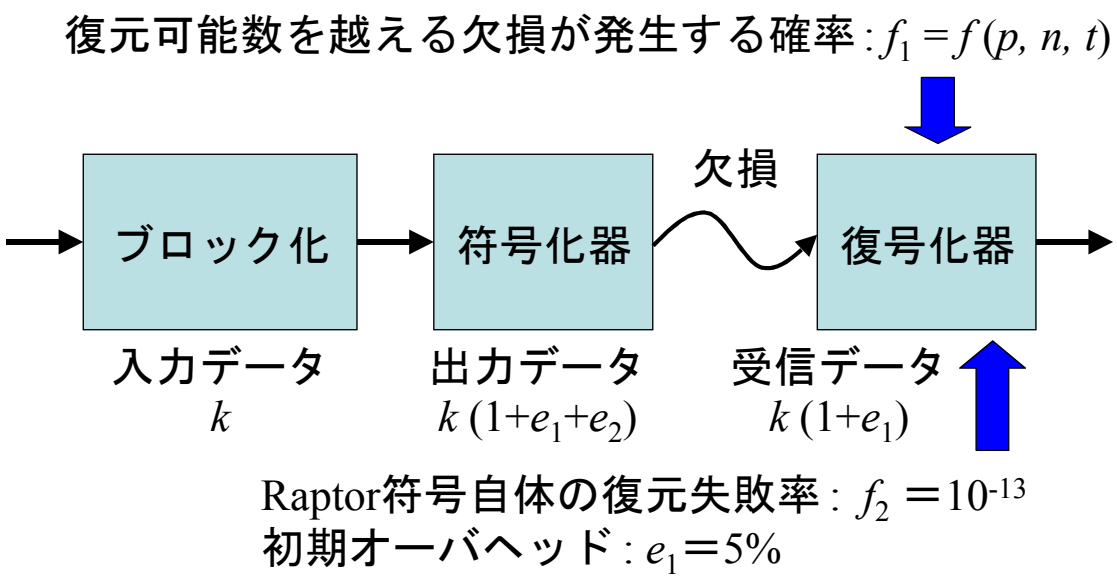


図 5-1-4-3　Raptor 符号を用いた伝送系

また、この伝送系全体での失敗率f_Tは、「ブロック内での欠損パケット数が復元可能数を超える確率(失敗確率)をf_1」「Raptor 符号での失敗率をf_2」とすると、

$$f_T = 1-(1-f_1)(1-f_2) = f_1 + f_2 - f_1 f_2 \cong f_1 + f_2$$

で与えられる。f_1 ならびにf_2が非常に小さい値の場合には、f_1 と f_2 の積は無視できる。

Raptor 符号では、出力シンボル数と入力シンボル数の関係によって、ブロック内で復元可能なパケット欠損数が決まる。つまり、伝送路で欠損が想定されるシンボル数に応じて、余分にシンボルを送出しておけば、受信側の復元に充分なシンボルを確保することが出来る。この関係は、出力パケット数と入力パケット数の関係と同等である。e_1 を初期オーバヘッド(伝送路にパケット欠損がない場合に必要なオーバヘッド), e_2 を追加オーバヘッド(伝送路のパケット欠損に対応して決まるオーバヘッド)とすると、出力パケット数／入力パケット数＝$1+e_1+e_2$ と表される。伝送路では想定されたパケット欠損が発生した場合、(出力パケット数－欠損パケット数)／入力パケット数＝$1+e_1$ という関係になり、受信側では初期オーバヘッド e_1 に相当する分だけ余計にパケットを受信すれば、ブロック内のすべてのパケットを復元することが(Raptor 符号自体の失敗率は存在しているが)可能となる。

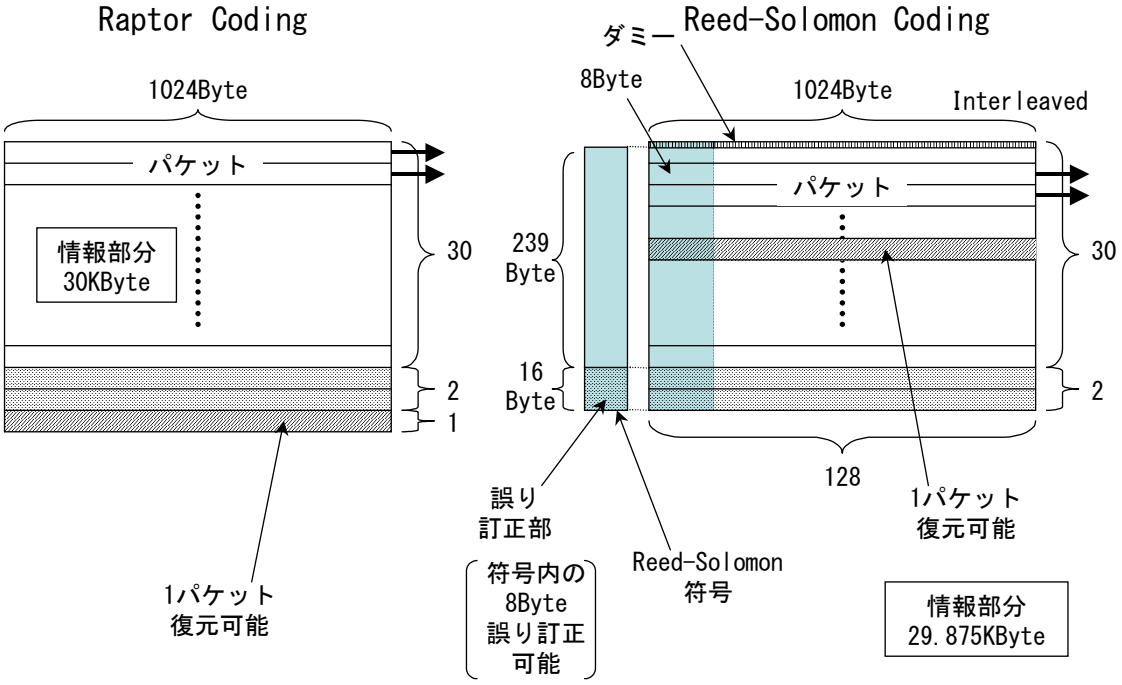


図 5-1-4-4　ブロック化の方法（１）

上記の考えに基づいて、送出ブロックのサイズを決める(図 5-1-4-4 参照)。映像のライブ配信を想定し、一定量のデータパケットを一定時間内に送出すると仮定する。Reed-Solomon 符号では、情報長 239Byte, 符号長 255Byte の符号語を想定し、これを 128 個集めて、インターリーブのためのブロックとする。元になる Reed-Solomon 符号は、8Byte の誤り訂正能力(復元能力)があり、この 8Byte を最小単位とするように、1024Byte のパケットとして送出する。この場合、1 パケットが欠損しても、各 Reed-Solomon 符号での 8Byte 欠損になるため、ブロック内の欠損パケットを復元できる。この場合、情報部分は 239Byte であり 8Byte で分割できないため、ブロック内にはダミーデータが必要となり、ブロック内の情報長は 29.875KByte となる。

Raptor 符号でも、情報長を Reed-Solomon とほぼ同じ値にするため、1024byte のパケットサイズで情報長 30KByte を送ると仮定した。この場合、情報長に相当するパケット数は 30 となり、5%の初期オーバヘッドとして 30 パケット×0.05＝1.5 パケットとなり、符号化時に 2 パケットを追加する必要がある。この状態では、欠損したパケットを復元できないので、想定されるパケット欠損に対応するパケットを追加する。図 5-1-5-4 の Reed-Solomon 符号は、1 パケット欠損しても、そのパケットを復元できる。Reed-Solomon 符号と同様のパケット欠損復元能力を与えるために、追加オーバヘッドとして 1 パケットを追加する。この場合、情報パケット数 32 に対して 1 パケットを復元できる Reed-Solomon 符号の方が、僅かに性能がよいことが分かる(表 5-1-4-2 参照)。

表 5-1-4-2　欠損パケットの復元数

	符号内のパケット数	復元可能パケット数
Reed-Solomon 符号 ＋インターリーブ	32	1
Raptor 符号	33	1

Reed-Solomon 符号は、生成多項式による演算を高速で行う必要があるために、ASIC や FPGA 等のハードウェア処理を行っている。そのため、Reed-Solomon 符号自体の誤り訂正能力を変更することは難しい。一方 Raptor 符号は、ソフトウェア的に欠損パケットの復元能力を高めることが出来る。図 5-1-5-4 に示すように、想定されるパケット欠損に応じて余分なパケットをブロックに追加することによって、欠損したパケットを復元できるのである。つまり、元のパケット数より 5%余計に出力パケットを受信出来れば、元のシンボルを復元することが可能となる。これは、符号化／復号化アルゴリズムがシンボル数に比例するという特性(線形性)を持ち、ネットワークの状況に応じてソフトウェア的に随時切り替えることが可能となる特性を利用したものである。

一方、Reed-Solomon 符号でも、以下の方法と併用して、欠損パケット復元能力を高めることが出来る。

(1) インターリーブの分散パターン変更

(2) 短縮化 (Shortning) 符号＋インターリーブ

上記の方法では、Reed-Solomon 符号化／復号化をする前に、ソフトウェア的にデータの組み替え等を行い、実質的にパケット復元能力を変更することが可能になる。この方法を用いた場合、Reed-Solomon 符号と Raptor 符号のパケット欠損復元能力を比較する。

5-1-4-4　インターリーブによるデータの分散

図 5-1-4-4 では、Reed-Solomo 符号を 8Byte 単位に分割しインターリーブをかけているが、これを 4Byte, 2Byte, 1Byte に変更したものが、それぞれ図 5-1-4-5～7 である。例えば図 5-1-4-7 のように 1Byte に分割すると、符号長は 255Byte であるため、255 個のパケットに分割される。これをそのまま 1024Byte のパケットとしてブロック化すると、ブロック全体のサイズが 255Kbyte(ダミーを含めると 256KByte)と、当初のブロックサイズの 8 倍になる。ブロックサイズが大きくなると、符号化／復号化のためのバッファサイズが増加するため、遅延が増大する。今回の評価はストリーミングを想定し、一定量のデータを一定時間に送出すると考えているため、パケットサイズを 1/8 にし、ブロックサイズを一定して計算を行う。ただ実際には、パケット長を変化させても、符号自体の復元能力は変化しない。

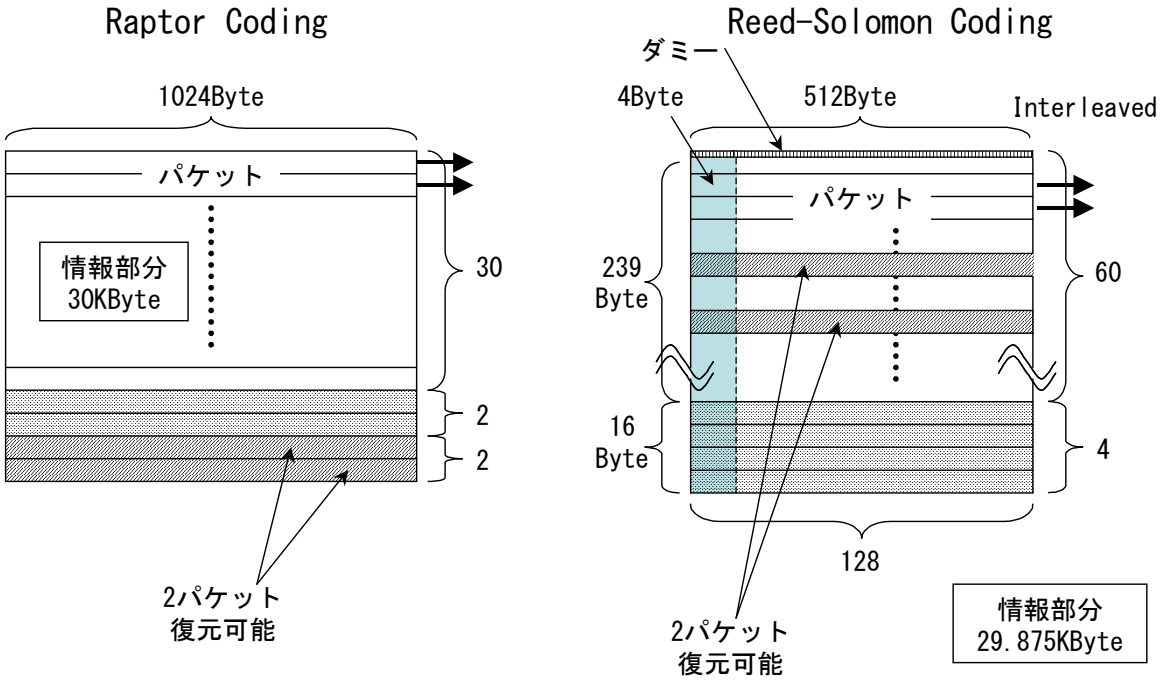


図 5-1-4-5　ブロック化の方法（2）

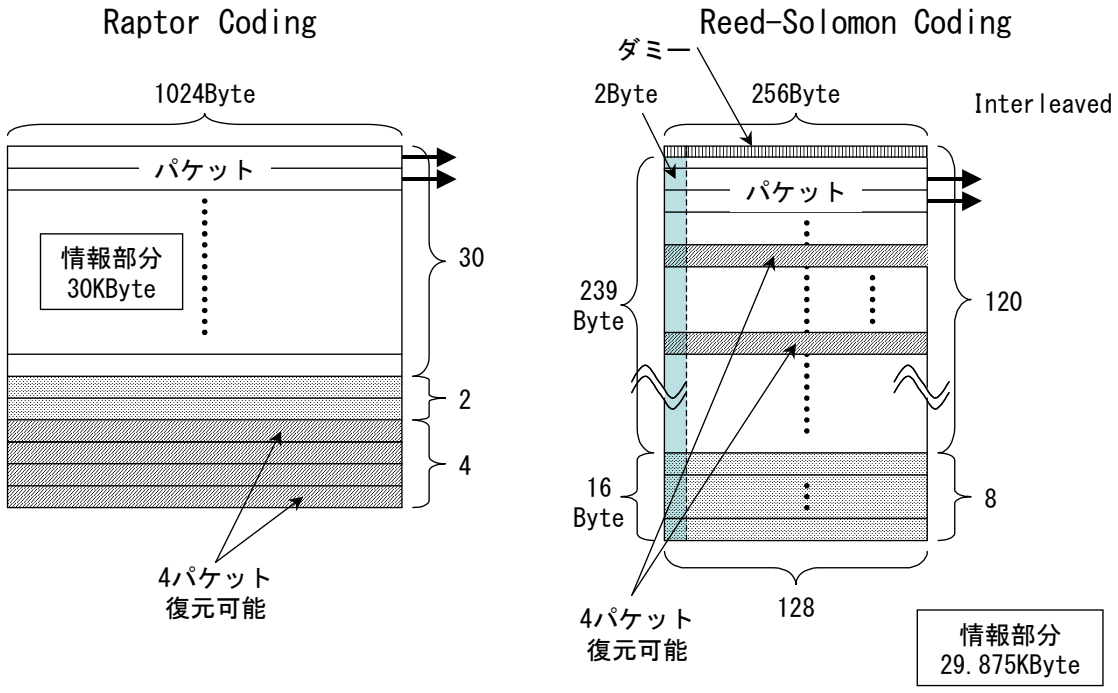


図 5-1-4-6　ブロック化の方法（3）

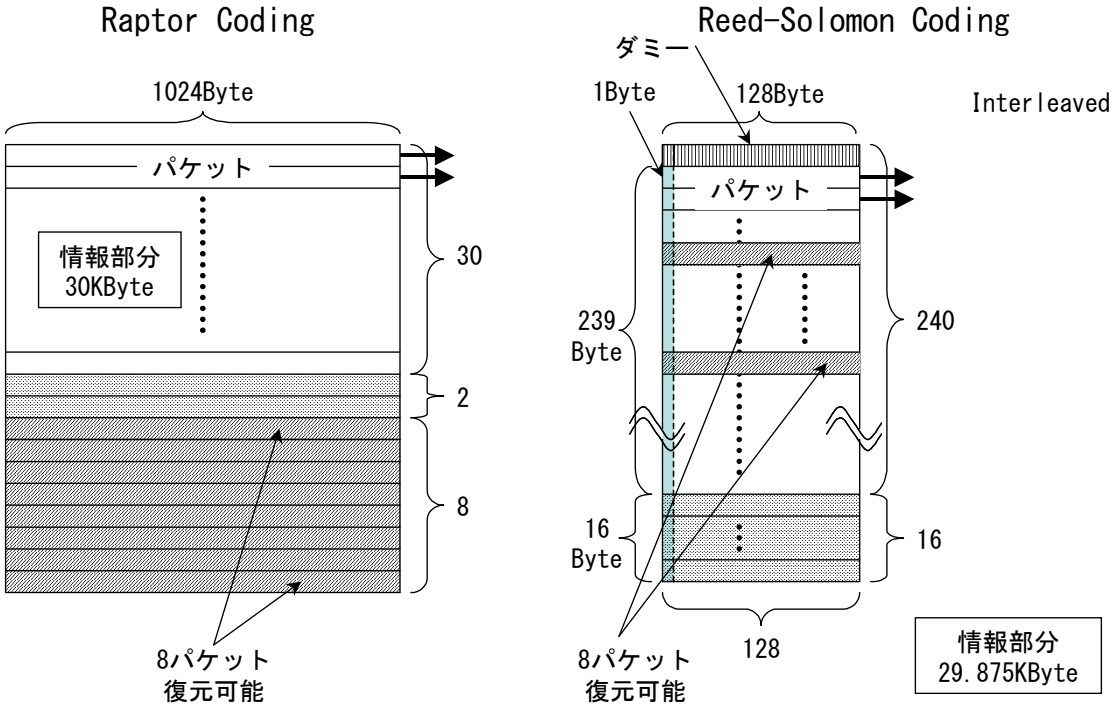


図 5-1-4-7　ブロック化の方法（４）

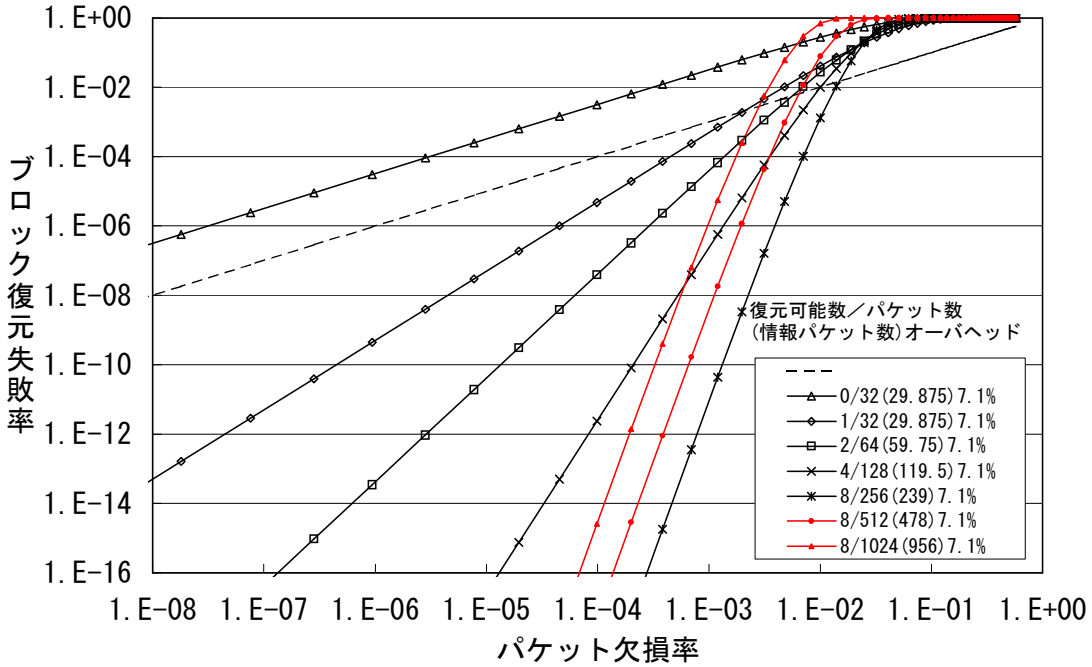


図 5-1-4-8　Reed-Solomon 符号のブロック復元能力

図 5-1-4-4～7 のようにインターリーブによってデータを分散させると、復元可能なパケット数は増加する。つまり Reed-Solomon 符号自体は、8Byte の訂正能力があるため、分割単位を 4Byte，2Byte，1Byte に変更する毎に復元可能個数は、2 個，4 個，8 個と増加する。ただ、分割単位を例えば 0.5Byte(4bit)にした場合、16 個の欠損パケットを復元することは出来ない。これは今回使用した

Reed-Solomon 符号自体が、バイト単位で最大 8 個の誤りを訂正できるからである。従って、Reed-Solomon 符号とインターリーブ法を組合せた場合、パケット復元能力に一定の上限があることが分かる。

図 5-1-4-8 は、Reed-Solomon 符号のブロック復元能力を示したものである。横軸が、伝送路で発生する平均的なパケット欠損率であり、これを図 5-1-4-4～7 のようにブロック化し、そのブロック中に発生したパケット欠損を復元した場合の失敗率を、縦軸に示してある。凡例で、[$t/n(k)$ s%]と表示されているが、この意味は以下に示す通りである。

t：ブロック内で復元可能なパケットの個数

（Reed-Solomon では、$t=(n-k)/2$ ）

n：ブロック内のパケット数 (符号長)

k：ブロック内の情報パケット数 (情報長)

s：$(n-k)/n\times 100$ (%)

図 5-1-4-8 では、「パケット欠損率」と「ブロック復元失敗率」が等しくなる場合の関係を、破線で示している。このグラフは横方向に２倍に拡大してあるため、本来傾き１になる直線が、傾き１／２になっている。この直線より必ず上にくる曲線[0/32(29.875)7.1%]は、ブロック化を行ってはいるが、欠損パケットの復元を行っていない場合の結果である。この場合「(平均的な)パケット欠損率：f_p」と「ブロック復元失敗率：f_B」が等しくなるような気がするが、３－４で述べたように、この両者の関係は、下記の式で規定される。つまり N_s=1 の時にのみ、この両者は等しくなる。

$$f_B = 1-(1-f_p)^{N_s}$$

図 5-1-4-8 のグラフと、図 5-1-5-4～7 の関係をまとめると、表 5-1-4-3 の様になる。図 5-1-4-4～7 に示されるように、復元可能なパケット数を増加させると、図 5-1-4-8 における「ブロック復元失敗率」は低下し、ブロック復元能力は向上していることが分かる。一方、復元に必要なオーバヘッドは、7.1%と一定の値である。本来この誤り訂正符号は、復元能力 t/n (3.1%)の以下のパケット欠損に対して、元のパケットを復元できる能力を持っている。ブロック内でのパケット欠損率が、復元能力 t/n (3.1%)より大きくなった場合に、ブロック復元に失敗することが分かる。復元能力 t/n (3.1%)より大きくなる割合は、二項分布に示されるように、ブロックに含まれるパケット数が増加することによって低減する。このため、復元能力 t/n の分母(つまりブロックに含まれるパケット数：符号長)が大きくなるほど、ブロック内でパケット欠損率が 3.1%を越える確率が低下し、見かけ上ブロック復元失敗率が低下する。

表 5-1-4-3　凡例の表示と対応する図番号

凡例の表示 [$t/n(k)$ s%]	対応する Reed-Solomon のブロック化
1/32(29.875) 7.1%	図 5-1-5-4(右)
2/64(59.75) 7.1%	図 5-1-5-5(右)

4/128(119.5) 7.1%	図 5-1-5-6(右)
8/256(239) 7.1%	図 5-1-5-7(右)

インターリーブ法を変更し、最小単位が 4bit 以下になるように分割しても、最大 8 個のパケットしか復元できないため、図 5-1-4-8 の[8/512(478) 7.1%]ならびに[8/1024(956) 7.1%]に示されるように、欠損パケット復元能力は低下する。

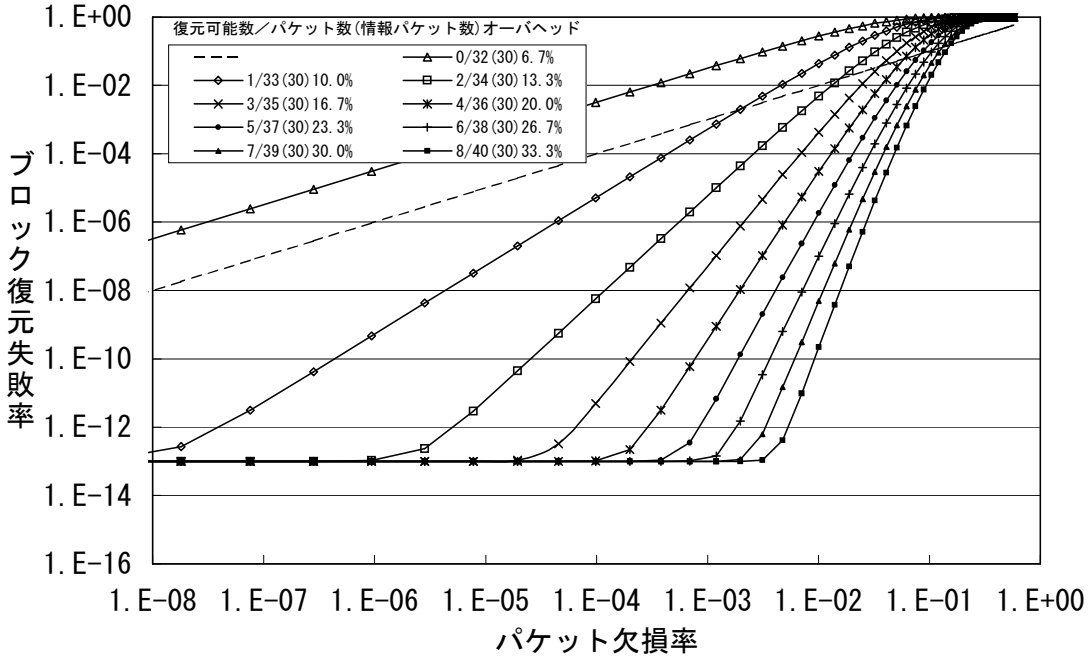


図 5-1-4-9　Raptor 符号のブロック復元能力（１）

一方 Raptor 符号では、欠損パケットに対応して任意のパケット数を追加できるので、図 5-1-4-4～7(左)に示す様に、誤り訂正能力を無制限に増加させることが可能となる。図 5-1-4-9，10 は、Raptor 符号のブロック復元能力を示したものである。横軸は、伝送路で発生する平均的なパケット欠損率であり、これを図 5-1-4-4～7 のようにブロック化し、そのブロック中に発生したパケット欠損を復元した場合の失敗率を、縦軸に示してある。凡例で、[$t/n(k)$ s%]と表示されているが、この意味は図 5-1-4-9 と同様である。Reed-Solomon との大きな違いは、Raptor 符号自体の持つ失敗率であり、図 5-1-4-9，10 では、10^{-13} のところに表れている。

表 5-1-4-4　凡例の表示と対応する図番号

凡例の表示 [$t/n(k)$ s%]	対応するブロック
1/33(30) 10.0%	図 5-1-4-4(左)
2/34(30) 13.3%	図 5-1-4-5(左)
4/36(119.5) 20.0%	図 5-1-4-6(左)

8/40(239) 33.3%	図 5-1-4-7(左)

表 5-1-4-4 は、凡例の表示と対応する図番号である。表 5-1-4-3 と比較すると、表 5-1-4-5 に示す様になる。つまり Reed-Solomon 符号では、オーバヘッド一定の状態(つまり、*t*/*n* が一定)で、インターリーブによってブロック復元失敗率を変化させているが、Raptor 符号では欠損パケット復元能力に応じて、オーバヘッドが増加し、直接、欠損パケット復元能力を高めている。

表 5-1-4-5　Reed-Solomon 符号と Raptor 符号の特徴

	Reed-Solomon 符号	Raptor 符号
復元可能個数 *t* と符号長 *n* の関係	*t*/*n* が一定	*n*-*t* が一定
情報長 *k*	一定	一定
オーバヘッド *s*	一定	*t* に対応して増加

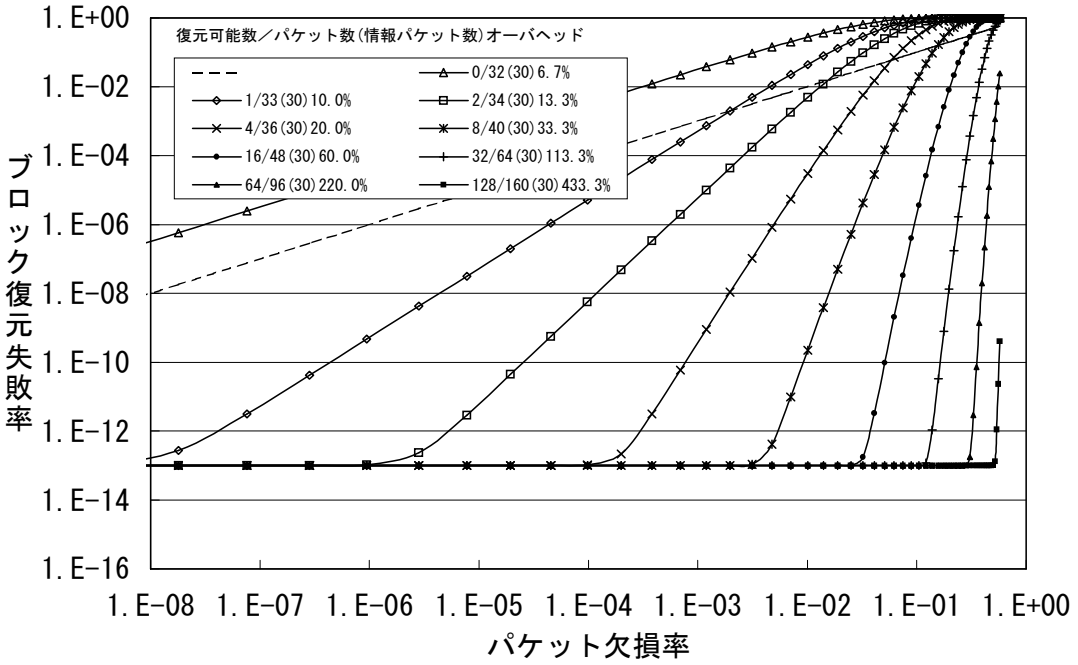


図 5-1-4-10　Raptor 符号のブロック復元能力（２）

図 5-1-4-10 に示すように、Raptor 符号においては、欠損が想定されるパケットを、無数に追加することによって、パケットの復元能力をいくらでも高めることが出来る。例えば、復元可能個数 *t* が 128 個の場合、パケット欠損率が数十%あっても 10^{-13} の復元失敗率で、パケット伝送が可能であることを示している。この結果から、パケット欠損復元能力を自由に変更することが可能な Raptor 符号は、性能的に遥かに優れていることが分かる。しかしオーバヘッドの大きい(符号化率の小さな)Raptor 符号と、オーバヘッドの小さな(符号化率の大きな)Reed-Solomon 符号を比較することは、不公平である。オーバヘッド(符号化率)

を考慮し、この両者を比較するために、オーバヘッドを考慮した実効的な欠損復元能力について考察する。

通常の誤り訂正符号では、*SN* 比を用いて比較する場合と、E_b/N_0 という値を用いる場合がある。E_b は、ビット当たりの電力であり、N_0 は雑音電力である(一般的な表記法として N_0 を用いるが、ここでは N_0=*N* である)。つまり、送出時のオーバヘッドを増加させるということは、送出するデータ量が増加することに相当し、ストリーミングの場合には、一定時間に一定量の情報部分を送出する必要があるため、オーバヘッドが増加した分だけ伝送速度を増加させる必要がある。

伝送速度の増加とビット当たりの電力との関係は、以下のように考えることが出来る。10Mbps データを高速の幹線系を経由して送信する場合、幹線系に誤り訂正機能を付加し、3 倍のデータ(つまりオーバヘッド 200%)で送出すると仮定する。この場合、幹線系で見ると、1bit 当たりの電力は同じであるが、このデータは幹線系の受信側で復号化され、3bit が 1bit に変換される。従って幹線系では、本来必要な 3 倍の電力が消費されていることが分かる。

この様に *SN* 比を捉えると、欠損パケットの復元は、2 つのステップで考えることが出来る。つまり、下記のような 3 つの場合である(図 5-1-4-11 参照)。

(1) 未符号化(ブロック化しない)の場合の、パケット欠損率(＝ブロック復元失敗率)

(2) 復元のために符号化したが、欠損を復元しない場合のブロック復元失敗率

(3) 復元のために符号化し、復元機能を用いて復元した場合の、ブロック復元失敗率

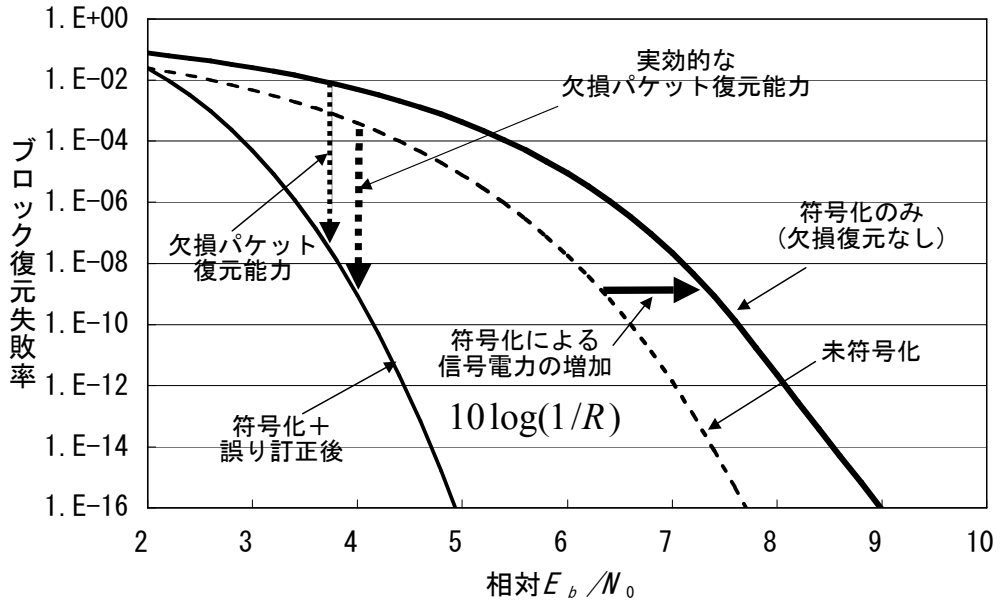


図 5-1-4-11　オーバヘッドを考慮したブロック復元失敗率

(1) 未符号化(ブロック化しない)の場合の、パケット欠損率 (＝ブロック復元失

敗率)

復元のための符号化を何も行わない場合のE_b/N_0とパケット欠損率のグラフを描く。この場合、$E_b/N_0=S/N$となる。パケットの欠損は、既に述べたように「伝送路でのノイズによる(パケット)損傷によるパケット廃棄」「中継器機(ルータ等)の輻輳によるパケット廃棄」の２つの原因で発生するが、この２つを等価的に扱い、すべて伝送路でのノイズによるものと便宜的に考える。この場合、等価的にブロックサイズ１と考えることが出来るので、パケット欠損率をブロック復元失敗率は、同じ値となる。

(2) 復元のために符号化したが、欠損を復元しない場合のブロック復元失敗率

符号化だけを行い、欠損復元を行わない場合のパケット欠損率を考える。この場合、符号化だけが行われるので、ビット当たりの電力である E_b の値が $1/R$ となる。ここで R は、符号化率であり、オーバヘッドとの関係は以下のように表される。

$$R = \frac{1}{1 + e_1 + e_2}$$

ビット当たりの電力は、$1/R$ となるため、パケット欠損率は変化せず、グラフが右に $\log(1/R)$だけ移動することが分かる。

(3) 復元のために符号化し、復元機能を用いて復元した場合の、ブロック復元失敗率

(2)で描かれたグラフに対して、欠損パケットの復元を行うと、ブロック復元失敗率も低下する。従って、グラフが下方向にシフトする。

この時の欠損パケット復元能力は、「最初に E_b/N_0 を基に描いたグラフ」と「最終的に得られたグラフ」との関係で示される。オーバヘッドを増加させることで欠損パケット復元能力を増大させても、オーバヘッド増加によるビット当たりの電力が増加すれば、その効果をうち消すことになる。従ってこの方法を用いると、異なるオーバヘッドを持つ２つの符号を比較することが可能となる。

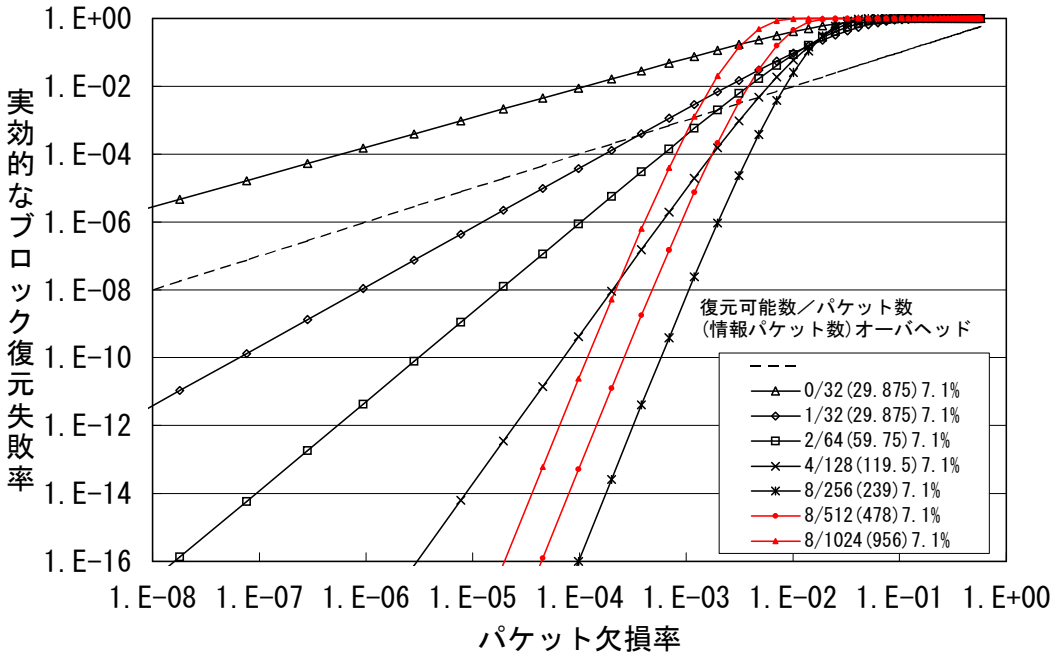


図 5-1-4-12　Raptor 符号の実効的なブロック復元能力

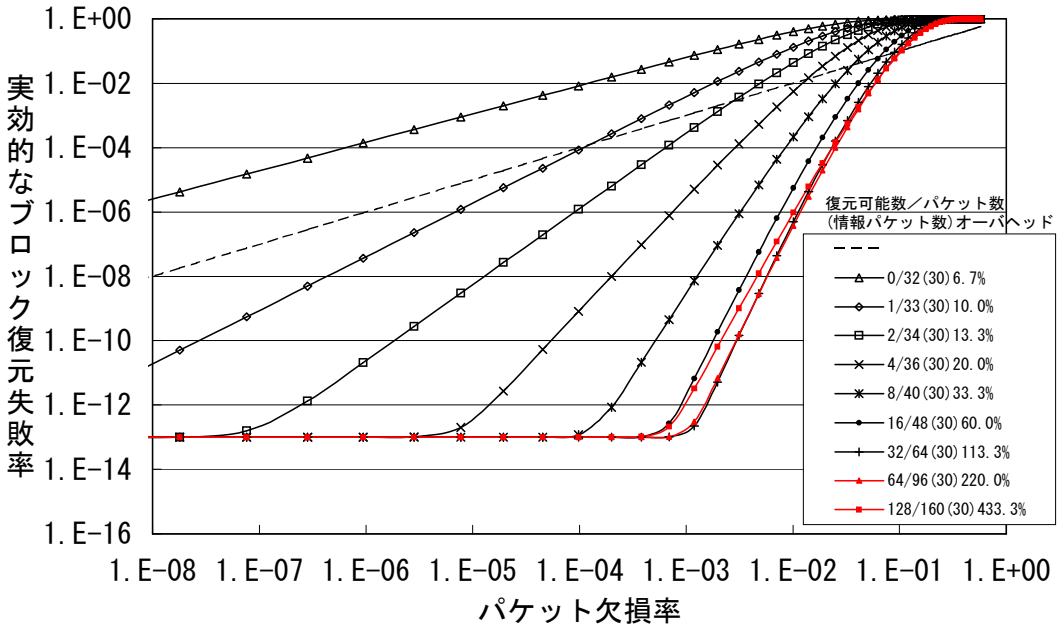


図 5-1-4-13　Reed-Solomon 符号の実効的なブロック復元能力

図 5-1-5-12 は、上記の考え方を基にして算出した「実効的なブロック復元能力」である。この結果を見ると、オーバヘッドを増大させるとある程度までは「実効的なパケット復元能力」が改善するが、それ以上改善しないことが分かる。つまり Raptor 符号には、実効的な欠損復元能力の限界があることが分かる。

同様の考え方で、オーバヘッド分だけ移動させた Reed-Solomon 符号の「実効パケット復元能力」を示したものが、図 5-1-5-13 である。図 5-1-5-12 と図 5-1-5-13

において、最も右の曲線が最も高いパケット欠損復元能力を有しているため、この両者を比較した。その結果パケット欠損復元能力は、3×10^{-4} 以上の高いパケット欠損率で、Raptor 符号の方が Reed-Solomon 符号より実効的なブロック復元失敗率が高いことが分かる。これ以下のパケット欠損が発生する領域では、Raptor 符号自体に復元失敗率があるため、復元失敗率の下限は 10^{-13} 以下にはならない。このため、ブロック復元失敗率の要求仕様が 10^{-13} を下回るような領域では、Raptor 符号を用いることは出来ない。

5-1-4-5　短縮化 (Shortning) 符号＋インターリーブ法

本来の誤り訂正符号から情報部分を取り除いて、別の誤り訂正符号を構成することが出来る。Reed-Solomon 等の符号を短縮化することによって符号長を変えることが可能であり、訂正可能なバイト数は変化しない。

基本とするのは(255, 239, 8)Reed-Solomon 誤り訂正符号で、順に符号長，情報長，誤り訂正能力を示す。この符号から、(128, 112, 8)，(64, 48, 8)，(32, 16, 8)を構成できる。この符号を基にブロック化を行い、ブロック復元失敗率を比較した(図 5-1-4-14)。その結果、欠損パケット復元能力自体は、パンクチャドによって、ある程度向上していくことが分かる。

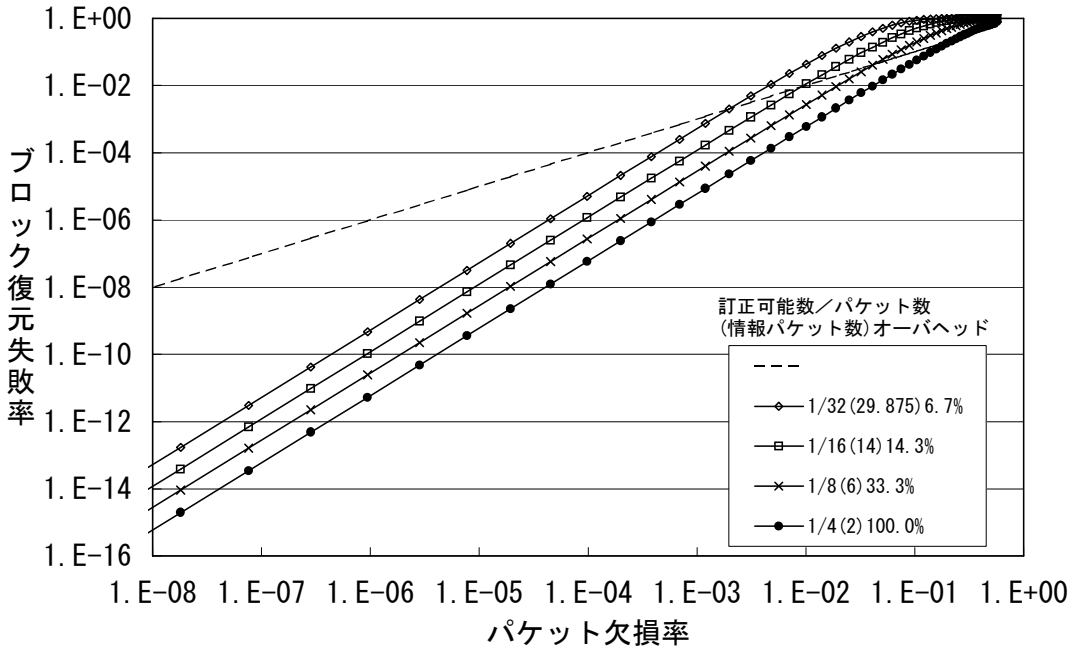


図 5-1-4-14　パンクチャドによるブロック復元能力

5-1-4-6　誤り訂正基本能力の比較

Reed-Solomon 符号と Raptor 符号の細かな違いは無視して、それぞれの符号を誤り訂正符号として捉え、その能力を比較してみる。つまり、Reed-Solomon 符号ではバイトではなく誤りの発生したパケットを訂正できると考え、Raptor 符号では、誤りの発生したパケットを修復すると考える。

一般的な誤り訂正符号の記述方法は、(*n*, *k*, *t*)である。誤り訂正能力を比較する場合、符号化率 *k*/*n* を横軸にとり、誤り訂正能力 *t*/*n* を縦軸にとる方法が用いられる。この場合の比較は、横軸である符号化率を一定にした場合の比較である。縦軸の *t*/*n* の値は、比の値として同じであっても、*t*/*n*=1/2 と *t*/*n*=100/200 という２つの誤り訂正能力は、その性能が大きく異なる。つまり、分母が大きいほど誤り訂正能力が高い。これは、先ほどのブロック化の例に示すように、ブロックサイズを大きくすることによってブロック内の誤り率が、伝送路の平均的な誤り率に近づくために、見かけ上の誤り訂正能力が上がるためである。また *t*/*n*=100/200 という誤り訂正能力を持つ符号は、100 個の連続的な誤りを訂正することが出来るが、*t*/*n*=1/2 という符号では、このような誤りを訂正する能力はない。この様な問題点はあるが、一般的な比較として、Reed-Solomon 符号と Raptor 符号を比較した。

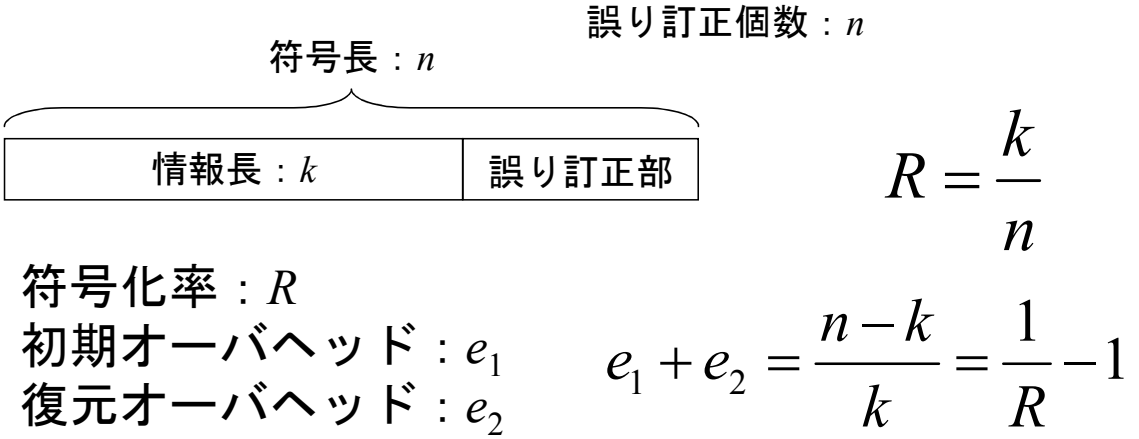


図 5-1-4-15　比較パラメータの定義

図 5-1-4-15 に、符号の性能比較に用いたパラメータをまとめる。Reed-Solomon 誤り訂正符号は、誤り訂正個数 *t* と情報長 *k* ならびに符号長 *n* との間に、下記のような「Reiger の限界式」が存在している。

$$t = \frac{n-k}{2}$$

Reed-Solomon 符号の誤り訂正能力 *t*/*n* は、以下のように表される。

$$\frac{t}{n} = \frac{(n-k)/2}{n} = \frac{1}{2}(1-\frac{k}{n}) = \frac{1}{2}(1-R)$$

また Raptor 符号の場合は、以下のようになる。

$$\frac{t}{n} = \frac{n-(1+e_1)k}{n} = 1-(1+e_1)\frac{k}{n} = 1-(1+e_1)R$$

つまり Reed-Solomon 符号は、誤り訂正用の冗長部分の半分の誤りしか訂正できない。この結果は、符号化率を 0 に近づけても全体の半分しか、誤りを訂正できないことを意味する。一方 Raptor 符号は、符号化率を小さくすることで、

符号全体の誤りを訂正することが出来る。ただ Raptor 符号には、初期オーバヘッドが存在し、この値によって符号化率が影響を受ける。

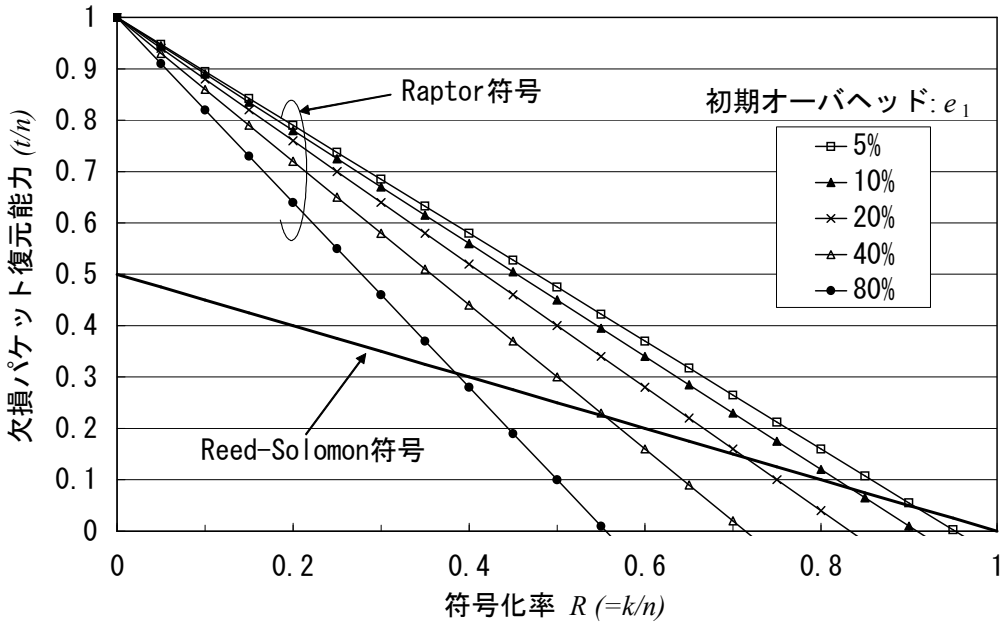


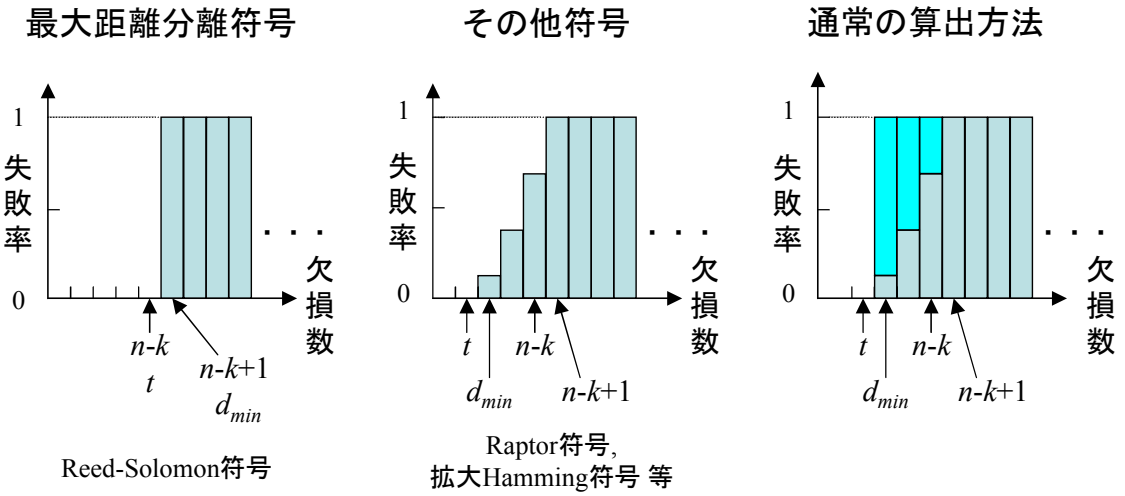
図 5-1-4-16　符号化率と誤り訂正能力の関係

図 5-1-4-16 は、Raptor 符号の初期オーバヘッドをパラメータとして、Reed-Solomon 符号と Raptor 符号の誤り訂正能力を比較したものである。Raptor 符号の初期オーバヘッドを 5%と仮定すると、符号化率が 90%以下では、Raptor 符号の方が、高い誤り訂正能力を有していることが分かる。逆に、初期オーバヘッドが大きくなると、Reed-Solomon 符号にとって有利な領域が拡大することが分かる。この結果から符号化率を 0 に近づけると、Raptor 符号の方が有利であることが分かる。ただ上記の比較は、ハードウェア化された誤り訂正用 Reed-Solomon チップを用いて、欠損復元を行った場合である。欠損復元用のソフトウェア Reed-Solomon を用いた場合には、Reed-Solomon 自体が Singleton 限界を満たす符号であるため、5%のオーバヘッドを必要とする Raptor 符号は、必ず下に来ることになる。

5-1-4-7　復元率の分布

これまでの計算では、最小 Hamming 距離から算出されて欠損復元能力以上のパケットが欠損すると、そのブロックの復元が失敗すると考えている。しかし実際には、符号語によっては他のどの符号語とも、最小 Hamming 距離以上離れている場合があるので、その符号語に関しては正しく復号出来る。図 5-1-4-17 は、その関係を示している。左の最大距離分離符号は、Singleton 限界を満たしているので、$t=n-k$ となる。この場合は、考え方がシンプルで、最小 Hamming 距離以上離れた場合、正しく復号することは出来ない(誤訂正となる)。従って、t 以下の欠損に対して失敗率は 0 となり、t+1 以上の欠損に対しては、失敗率は 1 となる。しかし、Singleton 限界を満たしていない符号に対しては、図 5-1-4-17

中央「その他の符号」の様に $t<n-k$ となり、失敗率は 0 と 1 以外の値を取る。一般的に符号の評価では、図 5-1-4-17 右「通常の計算方法」に示す様に、$t+1$ 以上の欠損が発生した場合には、すべて失敗(失敗率=1)と判断し、計算が行われている。しかし、図 5-1-4-17 中央の分布を用いて計算すると、計算精度を高めることが出来る。



n：符号長，k：情報長，d_{min}：最小ハミング距離，t：復元可能数

図 5-1-4-17　欠損復元能力の考え方

$$p \approx 0$$

$$f_S = \sum_{i=1}^{n} f_i \binom{n}{i} p^i (1-p)^{n-i} \approx f_1 \binom{n}{1} p^1 + f_2 \binom{n}{2} p^2 + f_3 \binom{n}{3} p^3 + f_4 \binom{n}{4} p^4 + f_5 \binom{n}{5} p^5 + \cdots + f_n \binom{n}{n} p^n$$

p 値が 0 に近いときには、欠損復元失敗率は、上記の様な式で表され、$t+1$ 次の多項式で近似できる。d_{min} から $n-k$ の欠損に対して 0, 1 以外の分布を与えると、f_S の値が低下し、復元失敗率が低下するが、f_S の次数は変化しない。従って、p の低次の項が存在すれば、その影響によって関数の次数が下がる。

従って、図 5-1-4-18 に示す様に、f_s の値は両対数グラフにおける傾き $t+1$ の直線として近似できることが分かる。

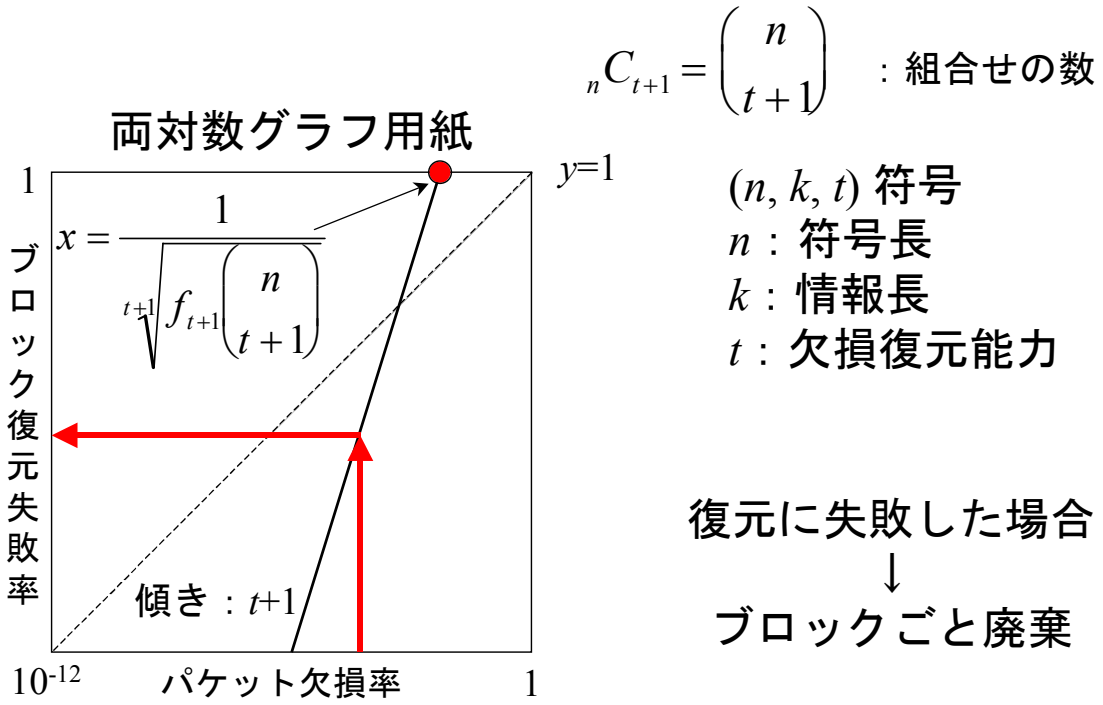


図 5-1-4-18　近似的なブロック復元失敗率の求め方

5-1-4-8　欠損復元パラメータの関係

表 5-1-4-6, 5-1-4-7 は、欠損復元符号の各パラメータの関係を示している。基本的には、S_{ES}, S_{EU}, S_{SB}, S_I, r_{ex}, R_{CB}, S_{IO}, n, S_{OO}, p, T_2, T_4 の各パラメータを決めれば、設計に必要な r_T, f_T, T_I の値を求めることが出来る。

このような関係をまとめることによって、符号語との欠損復元能力やオーバヘッドならびに遅延時間を、正しく評価することが可能となる。

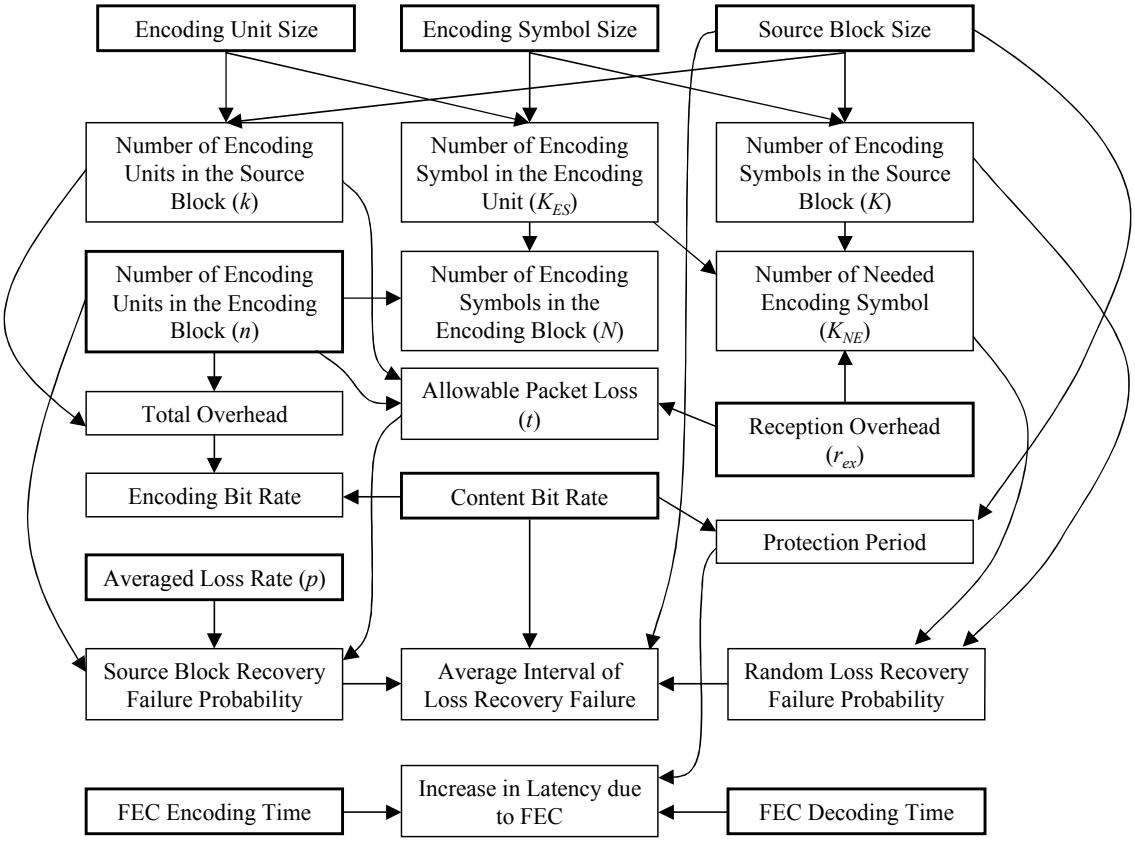


図 5-1-4-19　欠損復元符号のパラメータ関係図

表 5-1-4-6　欠損復元符号のパラメータ計算式

S_{ES}	Encoding Symbol Size (シンボルサイズ：符号語アルファベットのサイズ)	Byte	-
S_{EU}	Encoding Unit Size (Packet Payload Size) (エンコーディングユニットサイズ：パケットのペイロードサイズ)	Byte	-
S_{SB}	Source Block Size (a multiple of Encoding Unit Size) (符号化する際のソースブロックサイズ)	Byte	-
S_I	Information Size in the Source Block (ソースブロック中に含まれる有効な情報サイズ)	Byte	-
r_I	Information Ratio (ソースブロック内の有効な情報の割合)		$r_I = \frac{S_I}{S_{SB}}$
K	Number of Encoding Symbols in the Source Block (ソースブロックに含まれるシンボルの数)		$K = \frac{S_{SB}}{S_{ES}}$

k	Number of Encoding Units in the Source Block (ソースブロックに含まれるエンコーディングユニットの数)		$k = \frac{S_{SB}}{S_{EU}}$
K_{ES}	Number of Encoding Symbol in the Encoding Unit (エンコーディングユニットに含まれるシンボルの数)		$K_{ES} = \frac{S_{EU}}{S_{ES}}$
r_{ex}	Reception Overhead (初期オーバヘッド：復号時に必要な余分なシンボルの割合)		-
K_{ex}	Number of Extra Encoding Symbols beyond K needed to decode the Source Block (復号時に必要なシンボルの数)		$K_{ex} = \lceil r_{ex} K \rceil$
R_{CB}	Content Bit Rate (Playback Rate) 映像コンテンツの再生速度	bps	-
R_{IP}	Input Packet Number per second (Input Packet Throughput) (入力時のパケットの速度)	pps	$R_{IP} = \frac{R_{CB}}{8 S_{EU} r_I}$
S_{IO}	Input Packet Overhead (Packet Header) (入力時、ペイロードに付加されるヘッダサイズ)	Byte	-
R_{IS}	Input Stream Rate (入力時のストリーミングレート)	bps	$R_{IS} = 8 R_{IP} (S_{EU} + S_{IO})$
T_{PP}	Protection Period (保護時間：この時間内に発生した設定数のパケット欠損を復元できる)	msec	$T_{PP} = \frac{8 S_{SB} r_I}{R_{CB}} \times 1000 = \frac{8 S_I}{R_{CB}} \times 1000$
(k)	Number of Encoding Units in the Source Block		$k = \frac{S_{SB}}{S_{EU}} = \frac{K}{K_{ES}}$
k_{NE}	Number of Needed Encoding Units for Decoder (Needed Packet Number) (デコーダが復元に必要なパケットの数)		$k_{NE} = \left\lceil \frac{K + K_{ex}}{K_{ES}} \right\rceil \approx \left\lceil \frac{K + \lceil r_{ex} K \rceil}{K_{ES}} \right\rceil \approx \lceil k(1 + r_{ex}) \rceil$
K_{NE}	Number of Needed Encoding Symbol (a multiple of K_{ES}) (デコーダに渡されるシンボルの数)		$K_{NE} = k_{NE} K_{ES} = \left\lceil \frac{K + \lceil r_{ex} K \rceil}{K_{ES}} \right\rceil K_{ES} \approx \lceil k(1 + r_{ex}) \rceil K_{ES}$
n	Number of Encoding Units in the Encoding Block (Number of Output Packet)		-

	(エンコーディングブロックに含まれるパケットの数)		
N	Number of Encoding Symbols in the Encoding Block (エンコーディングブロックに含まれるシンボルの数)		$N = n\,K_{ES}$
r_T	Total Overhead (トータルオーバヘッド：符号化によって生じるオーバヘッド)		$r_T = \frac{n}{r_I k} - 1 = \frac{n - r_I k}{r_I k}$
S_{EB}	Encoding Block Size (エンコーディングブロックサイズ)	Byte	$S_{EB} = n\,S_{EU}$
R_{EB}	Encoding Bit Rate (出力時のエンコーディング速度)	bps	$R_{EB} = \frac{n}{k\,r_I} R_{CB}$
R_{OP}	Output Packet Throughput (Output Packet Number per second) (出力時のパケットスループット)	pps	$R_{OP} = \frac{R_{EB}}{8\,S_{EU}} = \frac{n}{k} R_{IP}$
S_{OO}	Output Packet Overhead (Packet Header) (出力時のパケットヘッダのサイズ)	Byte	-
R_{OS}	Output Stream Rate (出力時のストリームレート)	bps	$R_{OS} = 8\,R_{OP}(S_{EU} + S_{OO})$
t	Allowable Packet (Encoding Unit) Loss (復元可能な最大欠損数)		$t = n - k_{NE}$
d_{min}	(Equivalent) Hamming Distance (最大ハミング距離)		$d_{min} = t + 1$
p	Averaged Loss Rate (ネットワークの平均的な欠損率)		-
f_i	Failure Probability ($f_1, f_2, \bullet\bullet\bullet, f_n$) (欠損数に応じた復元失敗率)		$f_i = 0\,(1 \le i \le t)$ $0 \le f_i \le 1\,(t+1 \le i \le n-k)$ $f_i = 1\,(n-k+1 \le i \le n)$
f_S	Source Block Recovery Failure Probability (二項分布で与えられる復元失敗率)		$f_S = \sum_{i=1}^{n} f_i \binom{n}{i} p^i (1-p)^{n-i}$
f_R	Random Loss Recovery Failure Probability (符号自体が持つ失敗率)		$f_R(K, K_{NE})$
f_T	Total Failure Probability (上記２つの失敗率の合計)		$f_T = 1 - (1 - f_S)(1 - f_R)$ $\approx f_S + f_R$
T_A	Average Interval of Loss Recovery Failure (パケット欠損が発生する時間間隔)	day	$T_A = \frac{T_{PP}}{1000\,f_T} \div 60 \div 60 \div 24$
T_1	Buffering of Source Block at the Sender (符号化に必要なパケットが溜まる時間)	msec	$T_1 = T_{PP}$
T_2	FEC Encoding Time (Typical) (符号化に必要な時間)	msec	-
T_3	Buffering of Encoding Block at the	msec	$T_3 = T_{PP}$

	Receiver (復号化に必要なパケットが溜まる時間)		
T_4	FEC Decoding Time (Typical) (復号に必要な時間)	msec	-
T_L	Increase in Latency due to FEC (欠損復元機能を付加することによって追加的に必要な時間)	msec	$T_L = T_1 + T_2 + T_3 + T_4$

5-1-5　結論

誤り訂正符号として、Reed-Solomon符号を取り上げ、この符号と米国Digital Fountain社Dr. Lubyらにより開発されたLT符号について評価検討を行った。LT 符号は、Low-Density Parity-Check (LDPC) 符号の流れをくむ符号であり、復号のアルゴリズムが符号長の一乗に比例する。直接的に両者の誤り訂正能力を比較することは出来ないためReed-Solomon符号はインターリーブを用いると、下記のような点で、Raptor符号の方が有利であることが分かる。

(1) Raptor 符号の復号化アルゴリズムが、符号長に比例して処理時間が増加する。

(2) パケット欠損率が 3×10^{-3} 以上で、Raptor 符号の実効的な欠損パケット復元能力は高い。ただ、要求されるブロック復元失敗率が 10^{-13} であれば、3×10^{-3} 以下の領域でも、Raptor 符号が実用的に問題のないことが分かる。

(3) 処理時間が同等と考えられるハードウェア Reed-Solomon 符号とソフトウェア Raptor 符号を、同じ符号化率で比較すると、符号化率 90%以下で、Raptor 符号が有利となる。

今回の評価で、Digital Fountain社のRaptor符号(LT Codingの改良型欠損符号)が、有望であるという結果を得た。

なお本研究に関して、基礎的な部分は京都大学・高橋研究室に再委託を行った。再委託の研究成果報告書は、本報告書の最後に添付する。

5-2　ネットワークのモデル化とシミュレーション技術の開発ならびに高信頼性コンテンツ配信プロトコルの研究

5-2-1　序論

メタコンテンツ型の順方向誤り訂正技術を、所定のアプリケーションに対して、フロー毎にサービスレベルに応じて適用するネットワークモデルにおけるＱｏＳのシミュレーション技術の研究開発と検証を行なう。

ブロードバンドコンテンツのライブあるいはオンデマンド配信サービスを提供するネットワークシステムにおいて、順方向誤り訂正機能を有さないサーバ、順方向誤り訂正機能を有するサーバ、順方向誤り訂正機能を有さないクライアント、および、順方向誤り訂正機能を有するクライアントが混在する可能性のある、ヘテロジーニアスなネットワーク環境下において、所定のアプリケーションに対してフロー毎に、サービスレベルに応じた順方向誤り訂正機能を自律適応的に、あるいは、利用者の要求に応じて、選択的に適用・制御するためのプロトコル（パケット構造および通信シーケンス）案に関する、研究・開発をおこなう。

5-2-2　研究内容

多段ネットワークのモデル化を行う第一歩として、一段のネットワークモデルを考え、このシミュレーションを行っている。ネットワーク上にサポートサーバと呼ばれる機器を設置し、このサーバによってネットワーク上で発生したパケット損失を補うというモデルである。この結果を、電子情報通信学会全国大会で発表を行った。

また、Reed-Solomon 符号と DF 社の Raptor 符号の処理速度を比較し、その差を比較した。また、インターネット上で発生するパケット欠損とジッタの関係を調査し、その関係をモデル化することを試みた。

今後、これらのモデルにプロトコルを組み込んだ形での改良を行い、最終形態である自律適応型のフロー毎アプリケーション毎サービスレベル毎に必要な誤り訂正方法の検討を行う。なお本研究に関して、基礎的な部分は大阪大学・村上研究室に再委託を行った。再委託の研究成果報告書は、本報告書の最後に添付する。

5-2-3　誤り訂正符号の性能評価結果

5-2-3-1　冗長量に関する考察

ここでは､一般的なＦＥＣコードである Reed-Solomon と水平･垂直パリティ、および、メタコンテンツを用いた最新の Multi-stage 符号である Raptor の３つのアルゴリズムについての冗長量について考察する。なお、アルゴリズムにかけられる元のデータは、まずアルゴリズムの入力単位であるソースブロックに分割されていると仮定する。さらにこのソースブロックは、エンコードユニットという、Reed-Solomon、水平・垂直パリティ、Raptor のアルゴリズムにおけるエンコードの単位に分割されており、ソースブロック中のエンコードユニット数を k とすると

k = ソースブロックサイズ/エンコードユニットサイズ

が成り立つ。

以下の考察は、エンコードされた単位が一般的なパケットサイズで伝送されることを想定しており、各考察は以下の条件とする。

共通パラメータ：

エンコードユニットサイズ： 1024Byte 固定

k（ソースブロックサイズ中のエンコードユニット数）：

16, 32, 64, 128, 256, 512, 1024

これより、必然的にソースブロックサイズは(16KB、32 KB、64 KB、128 KB、256 KB、512 KB、1024 KB)となる。

Maximum Loss Provability in a Block（ソースブックサイズ内の最大パケットロス率)：ネットワークの平均パケットロスから、パケットロスの分布モデルをもとに算出され、k が小さくなるほど、大きな値をとる。

アルゴリズム固有のパラメータ

Reed-Solomon：

p：パケット廃棄率

m：k 個のエンコードユニットに対して付加する冗長符号数

水平・垂直パリティ：

v：垂直エンコードユニット数

h：水平エンコードユニット数

ここで、水平・垂直パリティアルゴリズムではエンコードユニットは長方形の形状にならべて水平・垂直それぞれにパリティを付与するため、ｋ＝ｖ×ｈが常に成り立つ。このため、(k,v,h) = (16,4,4)、(32,4,8)、(64,8,8)、(128,8,16)、(256,16,16)、(512,16,32)、(1024,32,32,)の条件について考察する。

5-2-3-2　冗長符号データサイズに関する考察

Reed-Solomon

冗長符号データサイズ（冗長符号数：ｍ×エンコードユニットサイズ= m KB)を計算し、これらをプロットした結果を図 5-2-3-1 に示す。

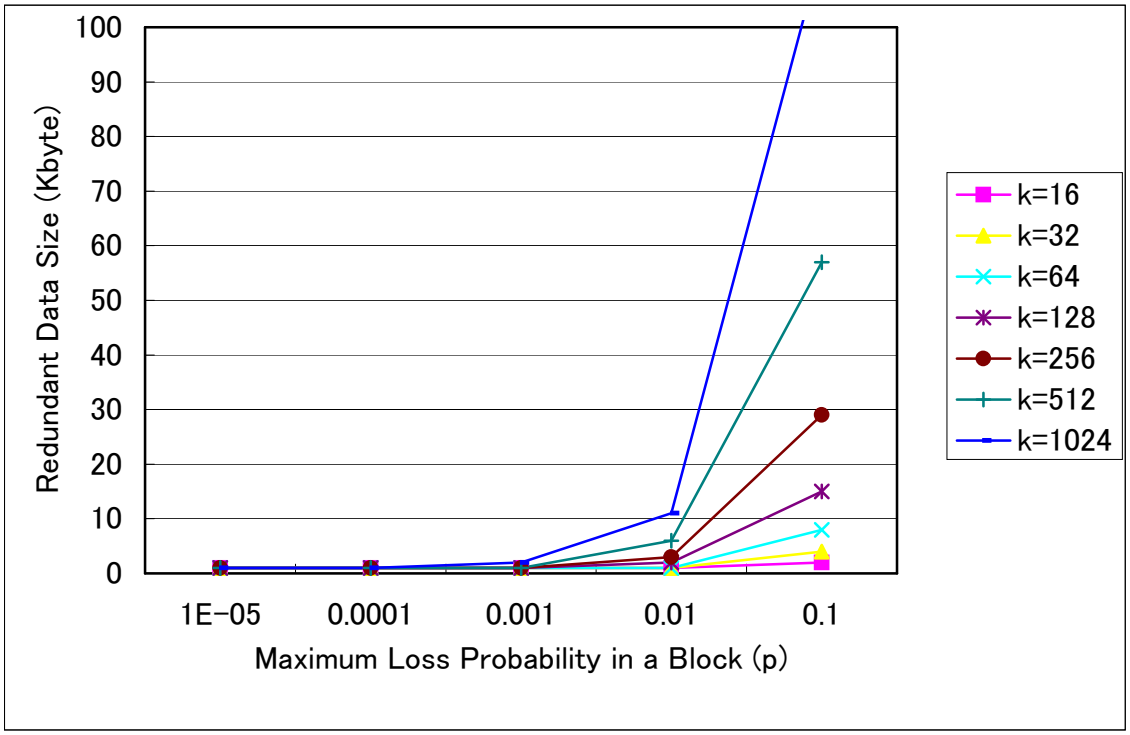


図 5-2-3-1　Reed-Solomon アルゴリズム のロス率に対する冗長量

Reed-Solomon アルゴリズムの場合、sender が送信する $k+m$個のパケットから receiver は任意の k 個を受信することでデータの復元が可能となる。したがって、送信するパケットが確率 p で廃棄された値（つまり 1-p が残る）が k以上であればデータは復元できる。よって、パケット廃棄率 p から最低限必要な冗長符号数mは

$(k+m)\times(1-p)=k$

よって

$m=p\times k/(1-p)$

となる。図 5-3-2-1 では上式で得られたmの ceiling（mより大きい最小の整数）を選択している。

水平・垂直パリティ

水平・垂直パリティの場合は、パラメータ（v, h）により、あらかじめ作成される冗長データサイズが決められているため、パケット廃棄率により冗長データが変化することはない。

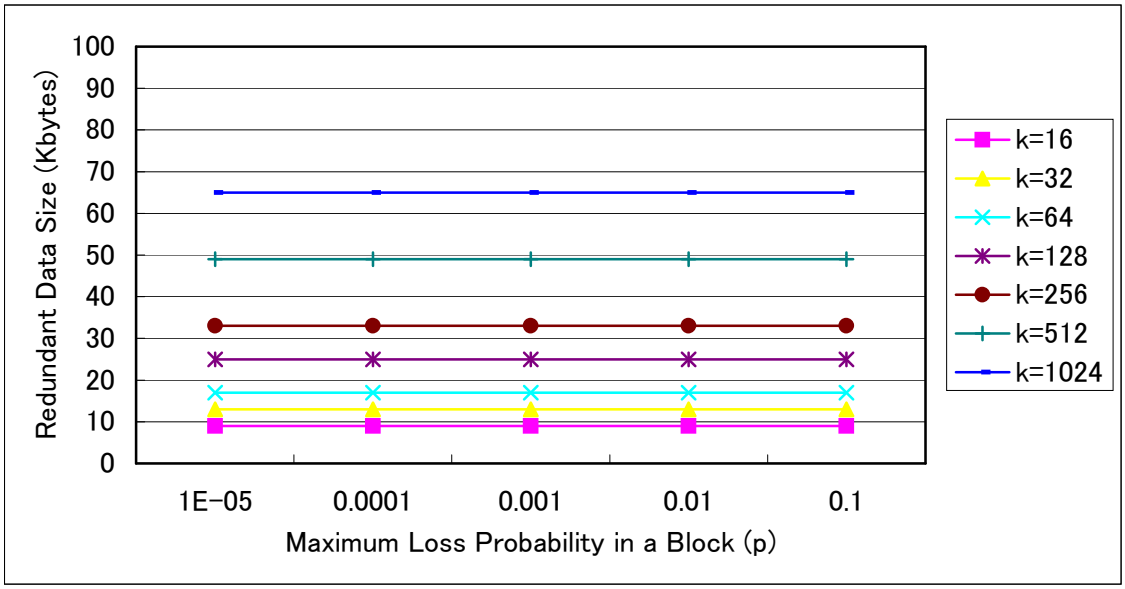


図 5-2-3-2　水平・垂直パリティアルゴリズムの冗長量

さらに、パケット廃棄率の値にかかわらずある行・列のパケット廃棄が 2 つ以上あるときに復元不可能な場合が存在するという性質を持つ。

図 5-2-3-2（$(k, v, h)=(9, 3, 3)$）の場合について説明する。図 5-2-3-3 の左では繰り返し、水平・垂直のパリティ計算をしていけば、デコード可能である。しかしながら図 5-2-3-3 の右では、２つ以上のパケット廃棄がある行・列を繋げると四角形ができており、互いにエラーを 1 追加にすることができず、情報を復元することは不可能である。これは（単一）パリティチェックは 2 重誤りを復元できないためである。

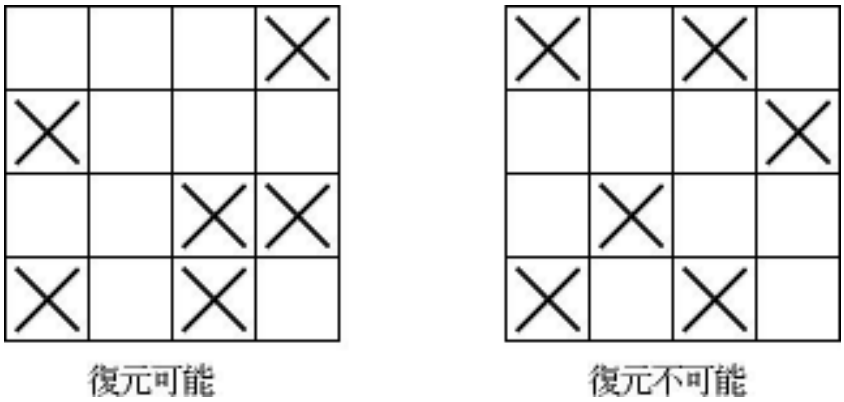


図 5-2-3-3　パリティエラー

Raptor

Raptor の冗長度は、デコードの失敗率に依存する冗長度とパケット廃棄率に依存する冗長度の積により計算される。前者は、たとえば Standard Mode の場合、パケット廃棄率 10^-11 のとき冗長度５％ 固定となる。後者は、ソースブロックサイズ内の最大パケットロス率により求まる。

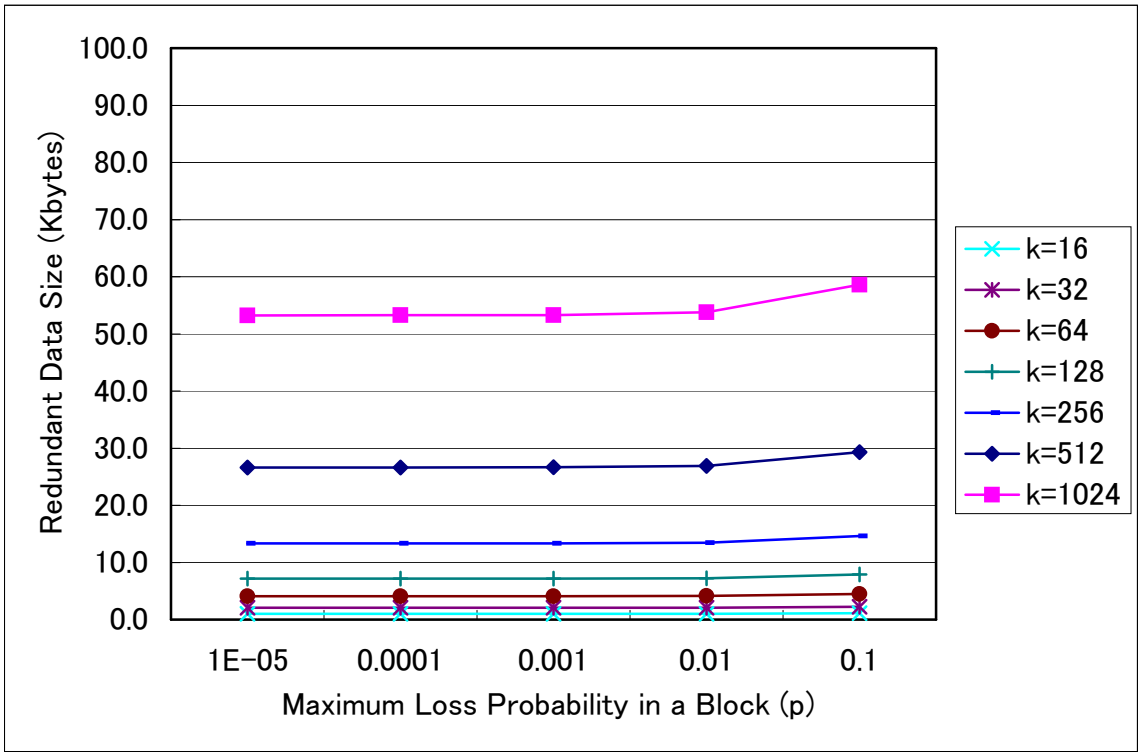


図 5-2-3-4　Raptor (Protocol-3 Standard Mode)アルゴリズムの冗長量

冗長量の比較

3 つのアルゴリズムの冗長量について比較する(図 5-2-3-5)。

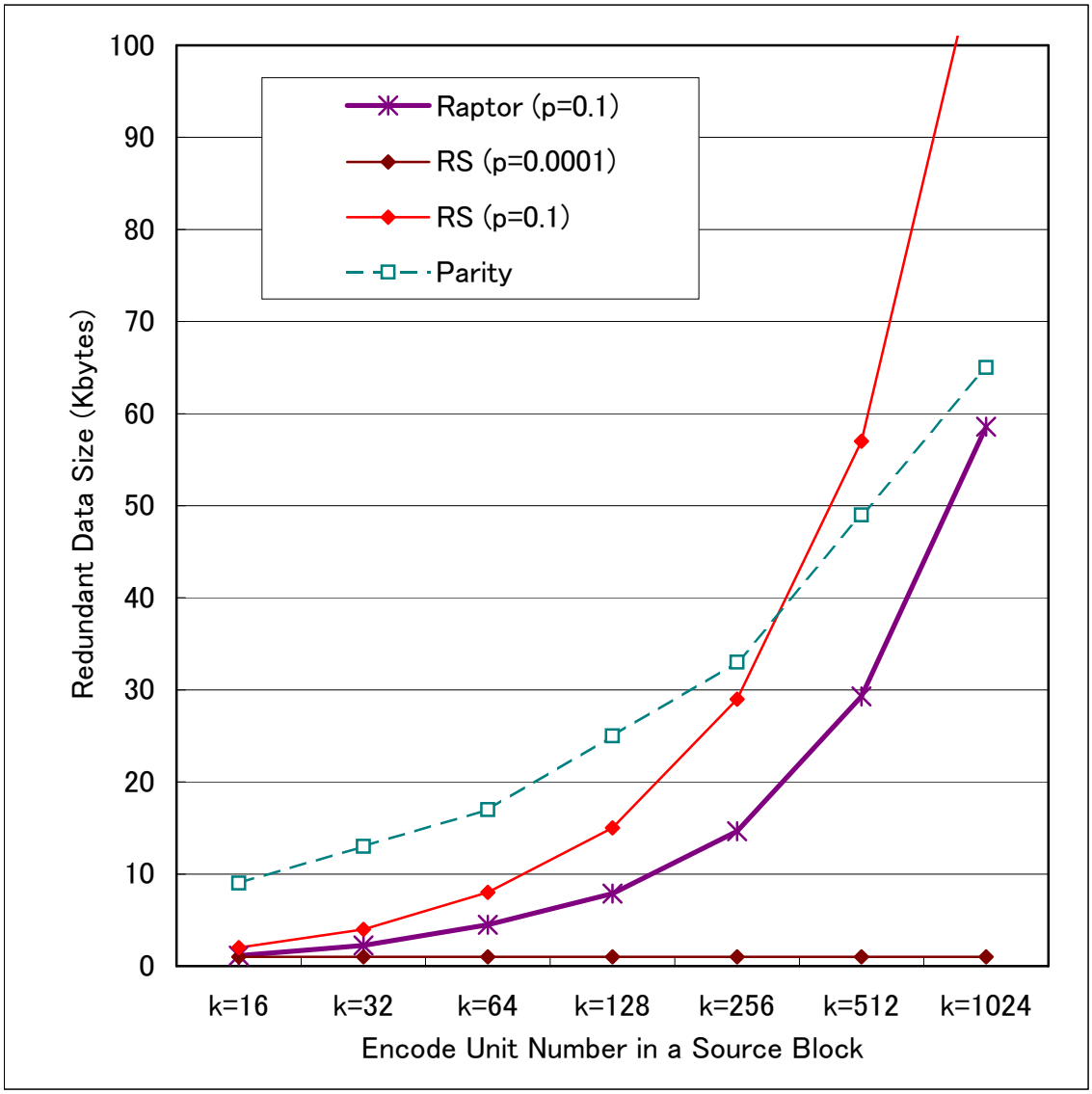


図 5-2-3-5　3 アルゴリズムの冗長量の比較

まず、Raptor、水平垂直パリティは、ブロック数の数によって決まった割合で付与するアルゴリズムであるため、エンコードユニットサイズが 1024byte と固定している状況では、ソースブロックサイズ中のエンコードユニット数 k が増加するに従い、線形に増加する。(ただし ceiling)。これに対して Reed-Solomon はパケットロス率に依存して冗長量が大きく変化するためアルゴリズムであるため、ロス率が低い状況では、k によらず低い値にとどまるが、ロス率が極端に高い領域では付与しなければならない冗長量が増加する。

この比較で注意すべきことは、それぞれのアルゴリズムの考え方の違いからデコードの失敗率が異なる、即ち、水平垂直パリティは大きなデコード失敗率を有する点である。

5-2-3-3　処理負荷の評価

5-2-3-3-1　概要および測定条件

各フォワードエラーコレクションアルゴリズムの相対的な処理負荷を比較するため、以下のような機器構成上で実験を行った。

表 5-2-3-1　機器構成

OS(*1)	Mandrake Linux 2.2.17-21mdk
CPU	Pentium III 933MHz
メモリ	768Mbyte

(*1) 実機検証で使用する MPEG-2 ストリーミング･エンジンを含むシステムでは、ストリーミングサーバは Windows-2000、デコーダは Linux ベースのシステム上で動作することが想定されているが、本章における評価では、エンコードとデコードの相対的な負荷を比較するため、同一の機器上で測定を行っている。

測定条件：

(1) 送信するソースデータのサイズは 1MByte に固定する（本測定ではランダムのテキストファイルを使用)。

(2) これに対して、エンコードユニットサイズは 1024byte 固定とし、k（ソースブロックサイズ / エンコードユニットサイズ)の値については、k=16, 32, 64, 128, 256, 512, 1024 の値で測定を行なう　(すなわち、ソースブロックサイズを 16Kbyte, 32Kbyte, 64Kbyte, 128Kbyte, 256Kbyte, 512Kbyte, および 1024Kbyte の範囲で変更し、それぞれの条件におけるデータのエンコード･デコード時間を測定する。

(3) それぞれの条件のもと、エンコーダとデコーダを同じ Linux ホスト上で動作させ、エンコーダにより符号化、あるいは冗長コードを負荷されたデータをソケット通信でデコーダへ転送し、デコーダ側でデータを復号する。

(4) 本章の実験では、ネットワーク上で実際にパケットロスが起こることは想定しない。すなわち全てのパケットが正常にデコーダへ到着する場合における処理負荷を測定する。

(5) ホストの負荷状況により、処理時間に揺らぎが生じるため、各実験ごとに最低十回の測定を行い、その単純平均を結果として採用する。

(6) 各アルゴリズムに対応するエンコード/デコード関数で要し CPU 時間を処理時間とする一従って通常ファイルまたはネットワークへのアクセスは処理時間に含めないが、処理の過程におけるバッファ間のメモリコピー等の操作等も符号化・復号化処理の一部と見なす。

5-2-3-3-2　Raptor アルゴリズム

本節では、Raptor アルゴリズムによる符号化、復号化処理時間の測定結果を示し、その考察を行う。

表 5-2-3-2 に、評価システム上で 1Mbyte のソースデータを転送した際、各条件におけるエンコード・デコード処理時間の平均を示す。

表 5-2-3-2　Raptor アルゴリズムによる
1MByte のソースデータに対する処理時間(単位:ms)

	k=16	k=32	k=64	k=128	k=256	k=512	k=1024
エンコード	0.94	0.71	0.62	0.31	0.31	0.17	0.10
デコード	0.28	0.18	0.10	0.07	0.05	0.03	0.03

これらの結果から、図 5-2-3-6 に示すように、ソースデータ全体に対する処理時間は、ソースブロックサイズとエンコードユニットサイズの比である k パラメータが小さい場合、(本実験ではエンコードユニットサイズを 1024 バイトに固定しているため、ソースブロックサイズが小さいことに相当する)、処理の負荷が大きくなる傾向にある。

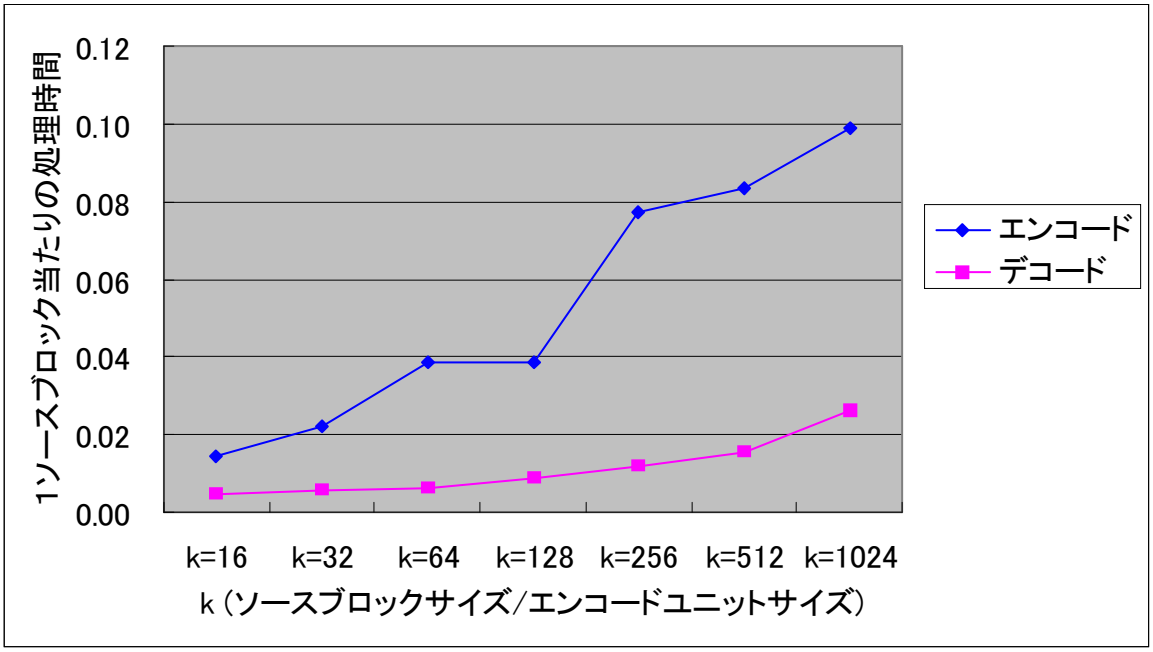


図 5-2-3-6　Raptor アルゴリズムによる
1 Mbyte のソースデータに対する処理時間　(単位:ms)

図 5-2-3-7 のように、Raptor アルゴリズムでは 1 ソースブロックあたりの平均処理時間は、そのソースブロックのサイズによる処理量の増加が後述する

Reed-Solomon アルゴリズムと比べ、非常に緩やかであることが分かる。これは、Raptor によるエンコード・デコード処理の計算量が小さいためと考えられる。

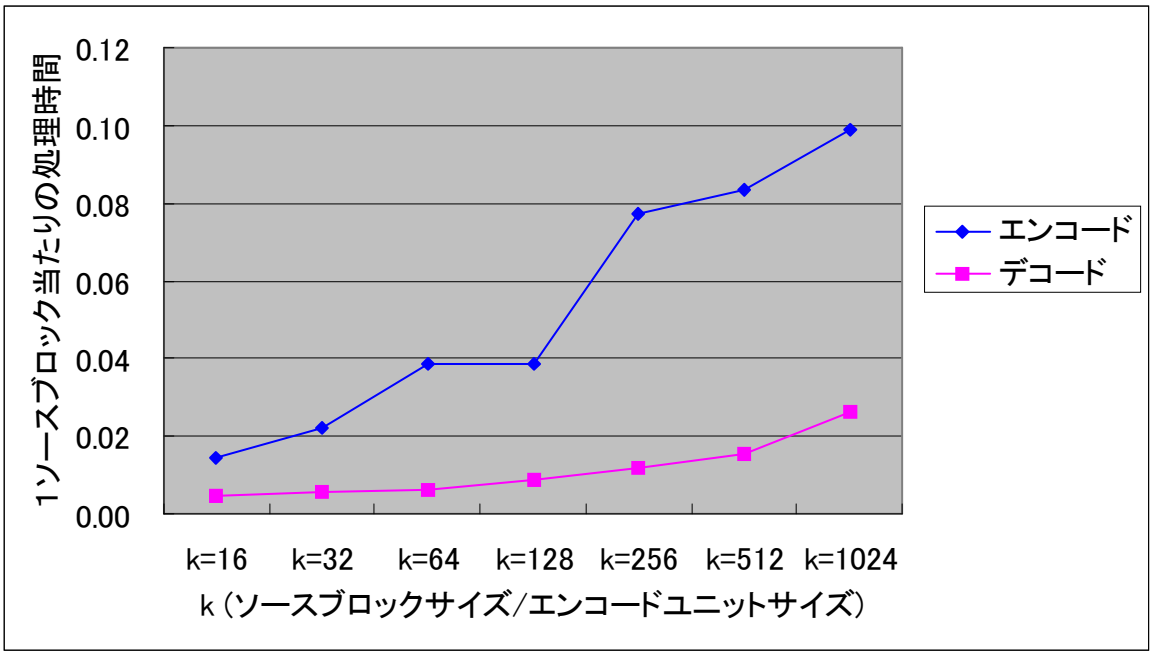


図 5-2-3-7　Raptor アルゴリズムによる 1 ソースブロック当たりの
平均処理時間（単位:ms）

5-2-3-3-3　Reed-Solomon アルゴリズム

本節では、Reed-Solomon アルゴリズムにおけるエンコード・デコード処理時間の測定結果を示し、これについて考察する。

Reed-Solomon アルゴリズムでソースブロックサイズ・エンコードユニットサイズのパラメータの他にネットワーク上の予想パケットロス率 p がパラメータとなる。本実験では m=0.00001, 0.0001, 0.001, 0.01, 0.1 について測定を行った。

エンコード時の処理時間の測定結果を表 5-2-3-3 および図 5-2-3-8 に示す。図に示すように、エンコード処理は Raptor と異なり、k の値が大きくなるに従い処理時間が急速に増加する。また、ソースブロック内最大パケットロス率 p が増加するにつれて、処理負荷が増加していく。

表 5-2-3-3　Reed-Solomon アルゴリズム(エンコード)による 1MByte の
ソースデータに対する処理時間（単位:ms)

エラーレート p	k=16	k=32	k=64	k=128	k=256	k=512	k=1024
0.00001	1.04	1.83	2.92	4.99	8.97	17.99	36.99
0.0001	1.05	1.85	2.95	5.15	9.21	19.02	39.25
0.001	1.06	1.84	2.99	5.34	9.61	19.40	57.88
0.01	1.08	1.85	3.02	8.18	24.02	77.85	269.30
0.1	1.56	4.30	13.50	46.46	175.84	665.75	2759.30

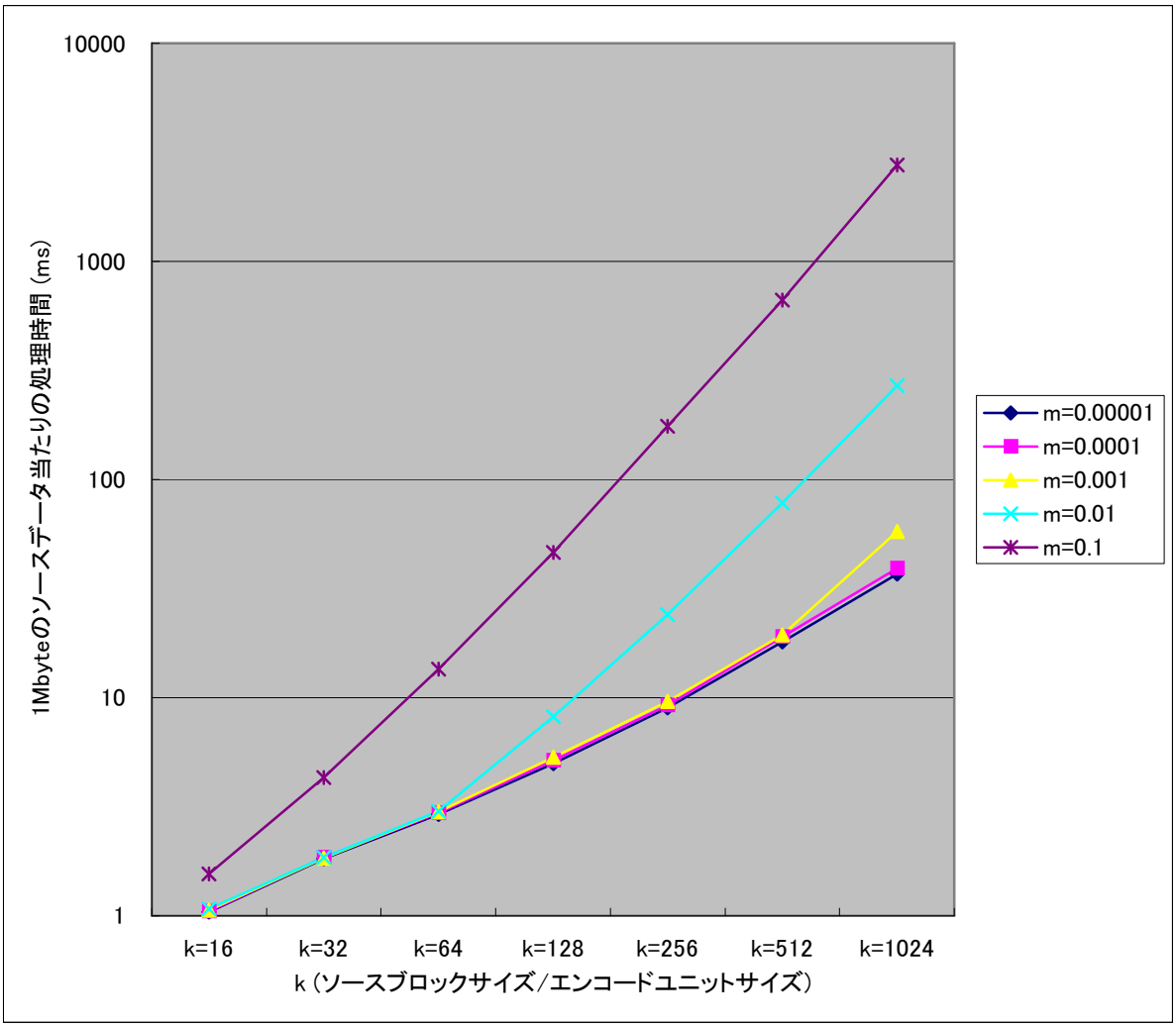


図 5-2-3-8　Reed-Solomon アルゴリズム(エンコード)による
1 Mbyte のソースデータに対する処理時間(単位:ms)

表 5-2-3-4 および図 5-2-3-9 に示すように、デコーダでもソースブロック内最大パケットロス率 p が増加するにつれて、処理負荷が増加していく。Raptor と異なり、k の値が大きくなるに従い処理時間が急速に増加する。

表 5-2-3-4　Reed-Solomon アルゴリズム(デコード)による

エラーレート	k=16	k=32	k=64	k=128	k=256	k=512	k=1024
p=0.00001	1.00	1.71	2.97	4.79	9.52	17.99	40.17
p=0.0001	1.00	1.78	2.99	4.99	9.59	18.20	39.18
p=0.001	1.00	1.80	3.02	4.83	9.94	18.27	66.30
P=0.01	1.02	1.73	2.92	7.49	23.20	68.12	291.60
P=0.1	1.55	4.02	14.24	45.70	200.20	650.06	2625.25

1MByte のソースデータに対する処理時間(単位:ms)

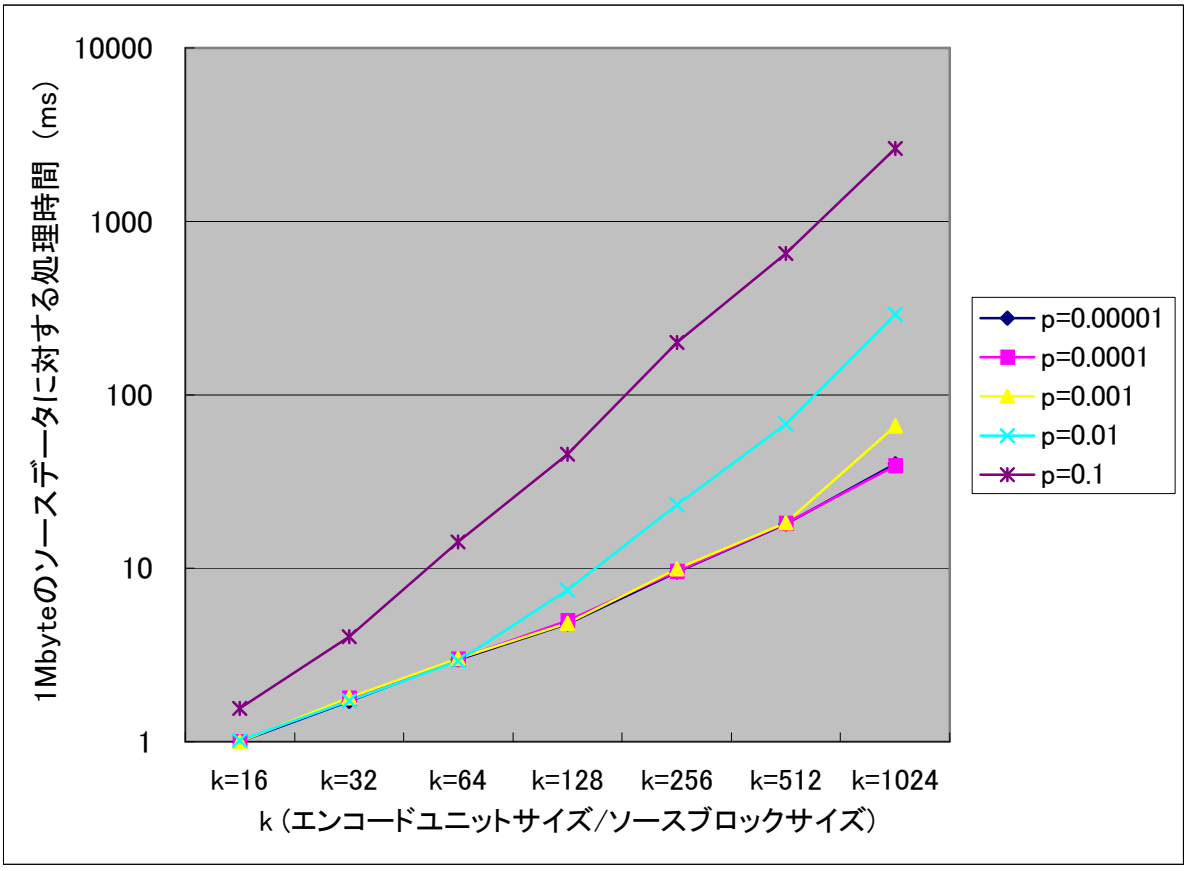


図 5-2-3-9　Reed-Solomon アルゴリズム(エンコード)による
1 Mbyte のソースデータに対する処理時間(単位:ms)

5-2-3-3-4　水平・垂直Parityアルゴリズム

本節では、水平・垂直パリティアルゴリズムによるエンコード・デコード処理時間の測定結果およびその考察を示す。パリティアルゴリズムではソースブロックサイズとエンコードユニットサイズの他に、水平方向のマトリックスのサイズ h および垂直方向のマトリックスサイズ v がパラメータとなる。

本実験では、v と h はソースブロックが正方形に近い形となるよう(すなわち h と v が近い値をとるよう)、各 k ごとに固定値を選択している。また、特にデコーダの処理負荷はパケットの損失数によって変化するが、ここでは本章の他のアルゴリズムに対する評価と同様、全てのパケットがデコーダに到着するという条件で測定を行なっている。

表 5-2-3-5　水平・垂直パリティアルゴリズムによる
1Mbyte のソースデータに対する処理時間　（単位: ms）

	(k, v, h)= (16, 4, 4)	(k, v, h)= (32, 4, 8)	(k, v, h)= (64, 8, 8)	(k, v, h)= (128, 8, 16)	(k, v, h)= (256, 16, 16)	(k, v, h)= (512, 16, 32)	(k, v, h)= (1024, 32, 32)
エンコーダ	0.35	0.22	0.21	0.16	0.16	0.15	0.15
デコーダ	0.28	0.2	0.17	0.15	0.16	0.14	0.14

これらの結果で明らかなように、水平・垂直パリティアルゴリズムは Raptor と同様、エンコード・デコード処理の負荷が比較的軽いため、k が小さい、すなわちソースブロックサイズが小さい場合は、バッファの切り替え・初期化などのオーバヘッドにより、処理時間が増加する傾向にあるが、k による処理時間の変化は Reed-Solomon に比べ小さい。ただし、水平・垂直パリティアルゴリズムは、本質的に全ての欠損パケットを救済できる訳ではない点に注意を要する。

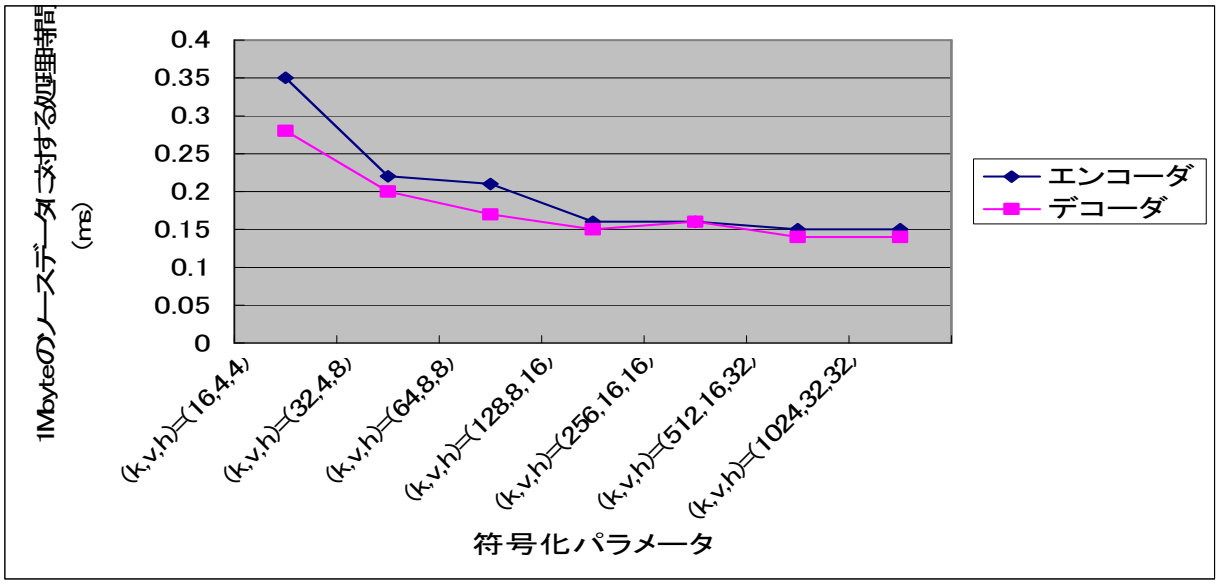


図 5-2-3-10　水平・垂直パリティアルゴリズムによる
1Mbyte のソースデータに対する処理時間（単位:ms）

5-2-3-3-5　各方式の比較

本節では、得られた測定データをもとに、Raptor, Reed-Solomon, Parity の三方式の比較を行う。負荷はパケットロス率により異なるが、ここでは p=0.0001 の場合について検討する。

エンコード時の各アルゴリズムの処理時間を表 5-2-3-6 に示す。

表5-2-3-6　各FECアルゴリズム（エンコード）による
1MByteのソースデータの平均処理時間の比較（単位:msec）

	k=16	k=32	k=64	k=128	k=256	k=512	k=1024
Raptor	0.94	0.71	0.62	0.31	0.31	0.17	0.10
RS	1.26	2.54	5.10	10.29	20.71	41.37	82.73
Parity	0.35	0.22	0.21	0.16	0.16	0.15	0.15

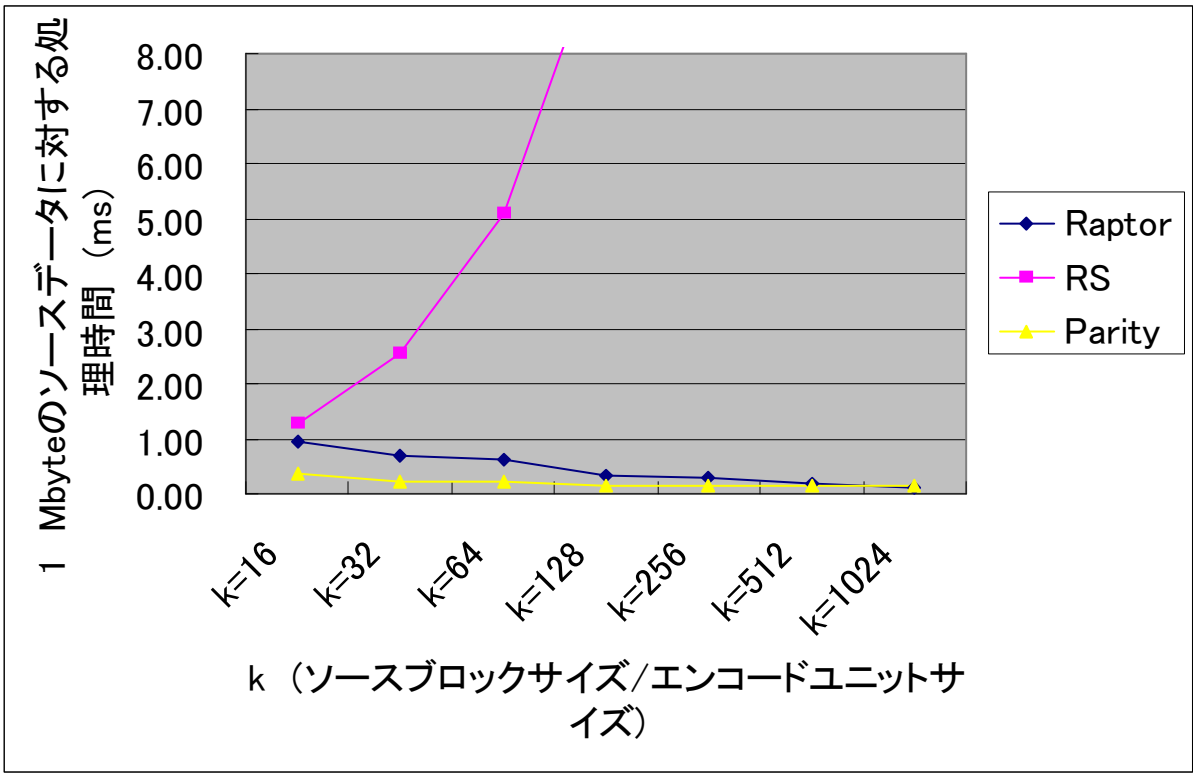


図5-2-3-11　各FECアルゴリズム(エンコード)による
1MByteのソースデータの平均処理時間の比較（単位:ms）

表 5-2-3-6 および図 5-2-3-11 の結果を比較すると、特に k=32～64 以上の値については Reed-Solomon のみ急速に処理負荷が増加することが分かる。一方、ParityおよびReed-Solomonアルゴリズムは、ほとんど変化しないことが分かる。

同様に表 5-2-3-7 にデコード時の各アルゴリズムの平均処理時間を示す。デコードの場合についてもエンコードと同様に、k の値が大きくなるにつれて Reed-Solomon アルゴリズムの処理負荷の増加が顕著となる。

表 5-2-3-7　各 FEC アルゴリズム（デコード）による
1MByte のソースデータの平均処理時間の比較（単位: msec）

	k=16	k=32	k=64	k=128	k=256	k=512	k=1024
Raptor	0.28	0.18	0.10	0.07	0.05	0.03	0.03
RS	1.00	1.78	2.99	4.99	9.59	18.20	39.18
Parity	0.28	0.2	0.17	0.15	0.16	0.14	0.14

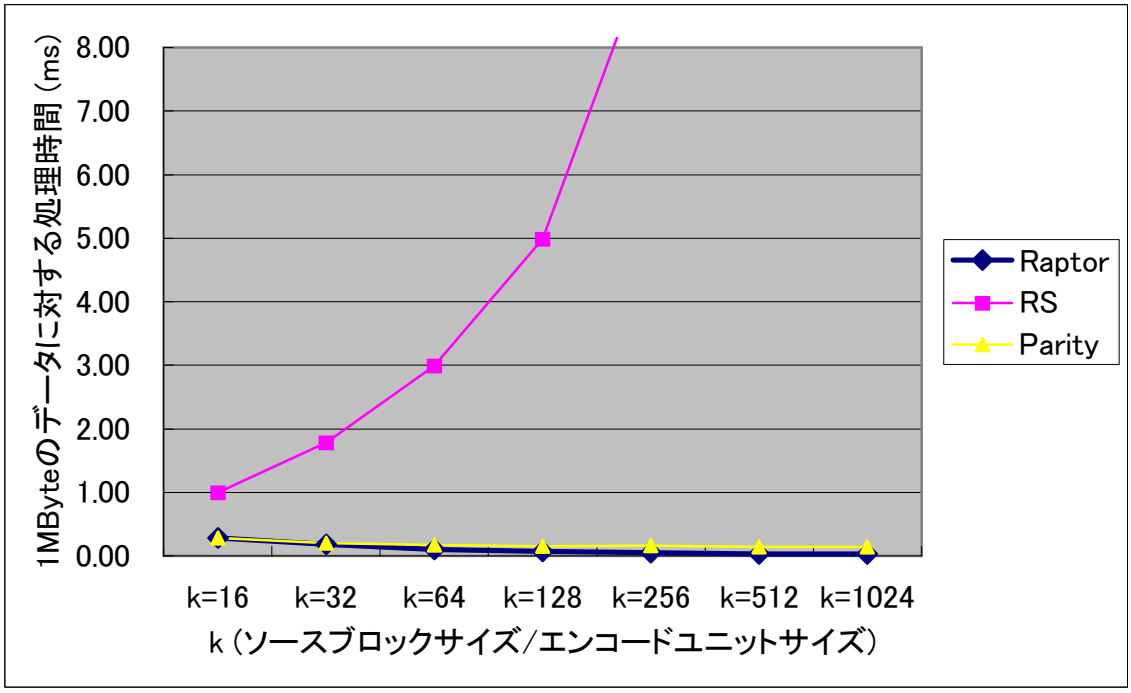


図 5-2-3-12　各 FEC アルゴリズム(デコード)による
1MByte のソースデータの平均処理時間の比較（単位:ms）

5-2-3-4　総合評価

以上の評価結果より、Raptor アルゴリズムは、Reed-Solomon より圧倒的に高速で、水平・垂直パリティーコード（バーストロスに弱い）並の高速処理性を有し、Reed-Solomon より僅かな冗長度で、広い範囲のパケットロス分布に対して、10^-11 オーダの失敗率で、パケット欠損の修復を行うことが出来る高性能なＦＥＣであると言える。

5-2-4-1　ネットワークのパケットジッタならびに欠損の測定

実際のインターネット環境で、パケットの欠損率とジッタ値を測定した。図 5-2-4-1 に、実験系の構成を示す。表 5-2-4-1 は、使用した機器の一覧である。まず 1028byte の UDP パケットを作成し、トラヒックジェネレータ(Sniffer 1)から 250msec 間隔で 10,000 個のパケットを送出した(約 32.896Kbps)。送出先は、USEN 側 BB ルータ(221.114.251.202)で、Ethernet ならびに光回線の伝送速度は 100Mbps である。

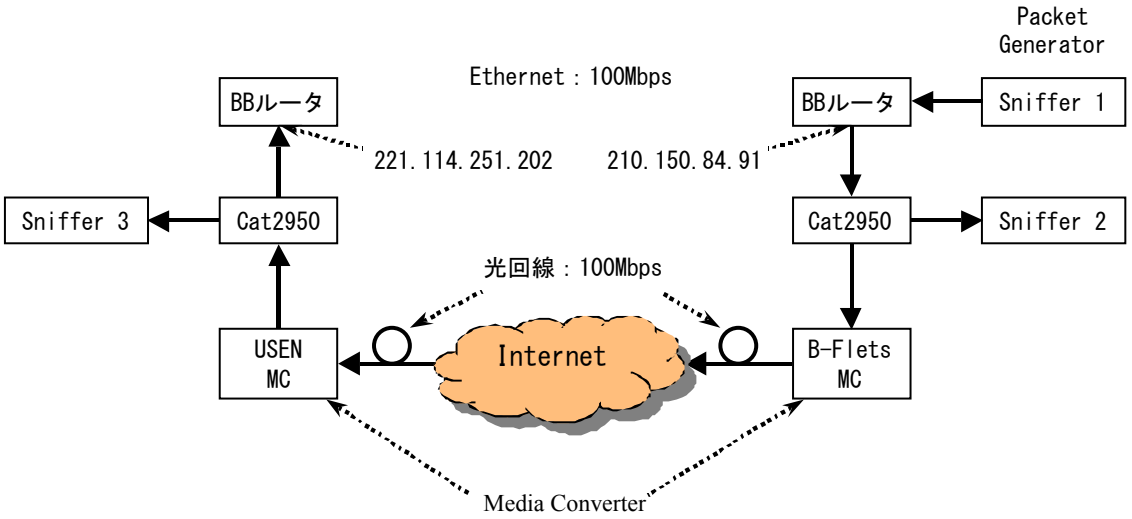


図 5-2-4-1　実験系の構成

表 5-2-4-1　使用した機器

	機 器 名 称
BB ルータ	Corega BAR SW-4P Pro (10BASET/100BASE-TX に対応した WAN ポートと 4 ポートのスイッチング HUB)
Cat 2950	Cisco Catalyst 2950 (WS-C2950-24) ワイヤスピード：3.6Mpps
B-Flets MC	EX 型光加入者線終端装置 (東仕 110004 号 1 版)
USEN MC	OMC-3201FX/SSM-A-1310

図 5-2-4-1 の実験系で、Sniffer 2 ならびに Sniffer 3 でのパケット到着時刻を測定し、この両者を比較することによって、Internet のパケット欠損率ならびにジッタ値を測定した。Sniffer1 からは、10,000 個のパケットが送出されるため、Sniffer 3 でのパケット到着数と比較すると、Internet でのパケット欠損数が分かる。今回の測定では、65 個のパケットが欠損し、欠損率は、0.65%であることが分かる。

図 5-2-4-2 は、ジッタを測定する方法を示したものである。パケットの出発時刻(Sniffer 2 への到着時刻)を縦軸に取り、パケットの到着時刻(Sniffer 3 への到着時刻)を横軸に取る。ジッタがなく、出発時間と到着時間を測定する装置のクロックが一致していれば、両者の関係は傾き 1 の直線で表される。ジッタがあると、結果は折れ線グラフとなる。この折れ線グラフから、一次の最小自乗法を用いて理想的な到着直線を算出する。この直線の傾きは、クロックの

違い等によって 1 からずれる。実際の到着時刻と理想的な到着直線からの横方向のずれが、遅延時間(ジッタ)を表す。この折れ線の傾きは、(パケットが等間隔で送出されている場合には)伝送速度の大きさに比例し、傾きが緩やかな場合は、伝送速度が低下していることを示す。

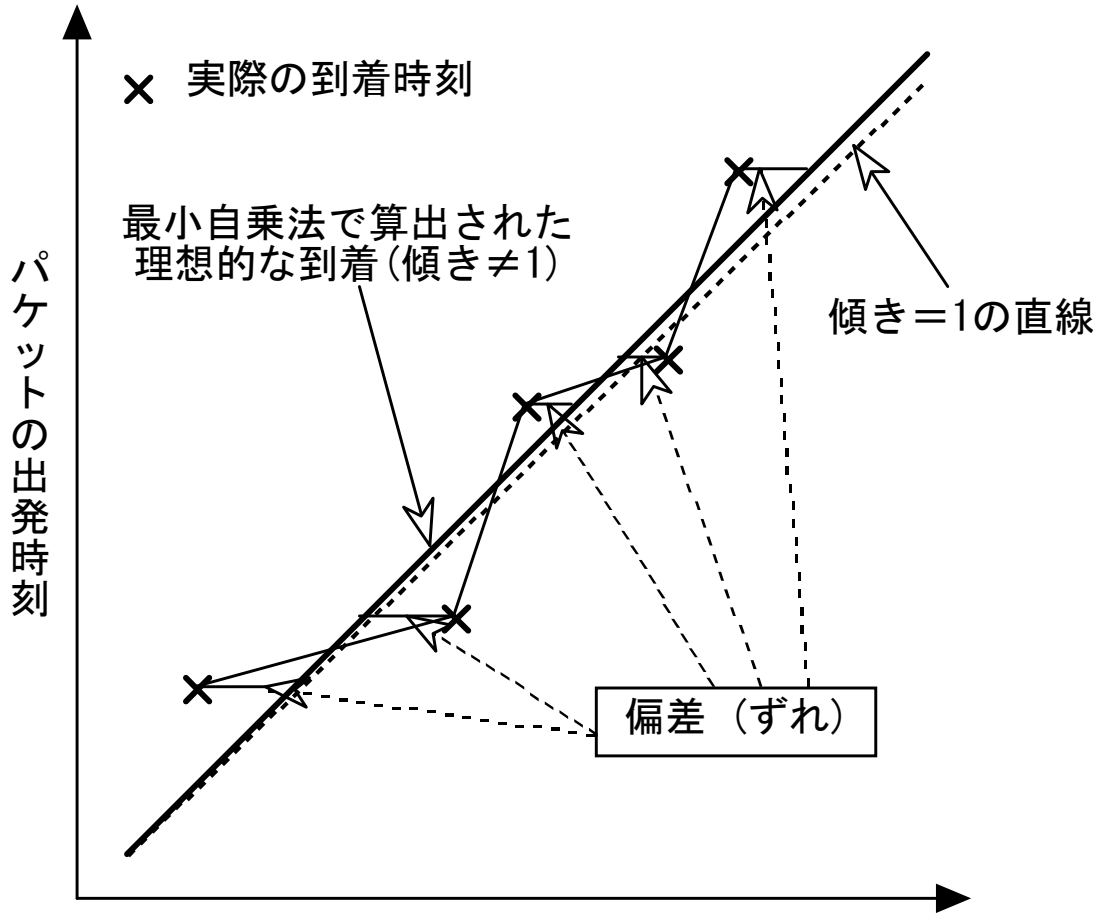


図 5-2-4-2　ジッタの測定方法

このグラフでは、横方向のずれである遅延時間をグラフ上に表現することが難しいので、その値を差し引いて図 5-2-4-3 の様に、縦軸をジッタ値に変更する。今後は、このグラフを用いてジッタを評価する。到着時刻の測定に Sniffer 3 を用いているため、±0.6msec 程度の測定誤差が発生するが、パケットの送出間隔が 250msec と大きいので､実験結果に大きな影響を与えないと考えられる。

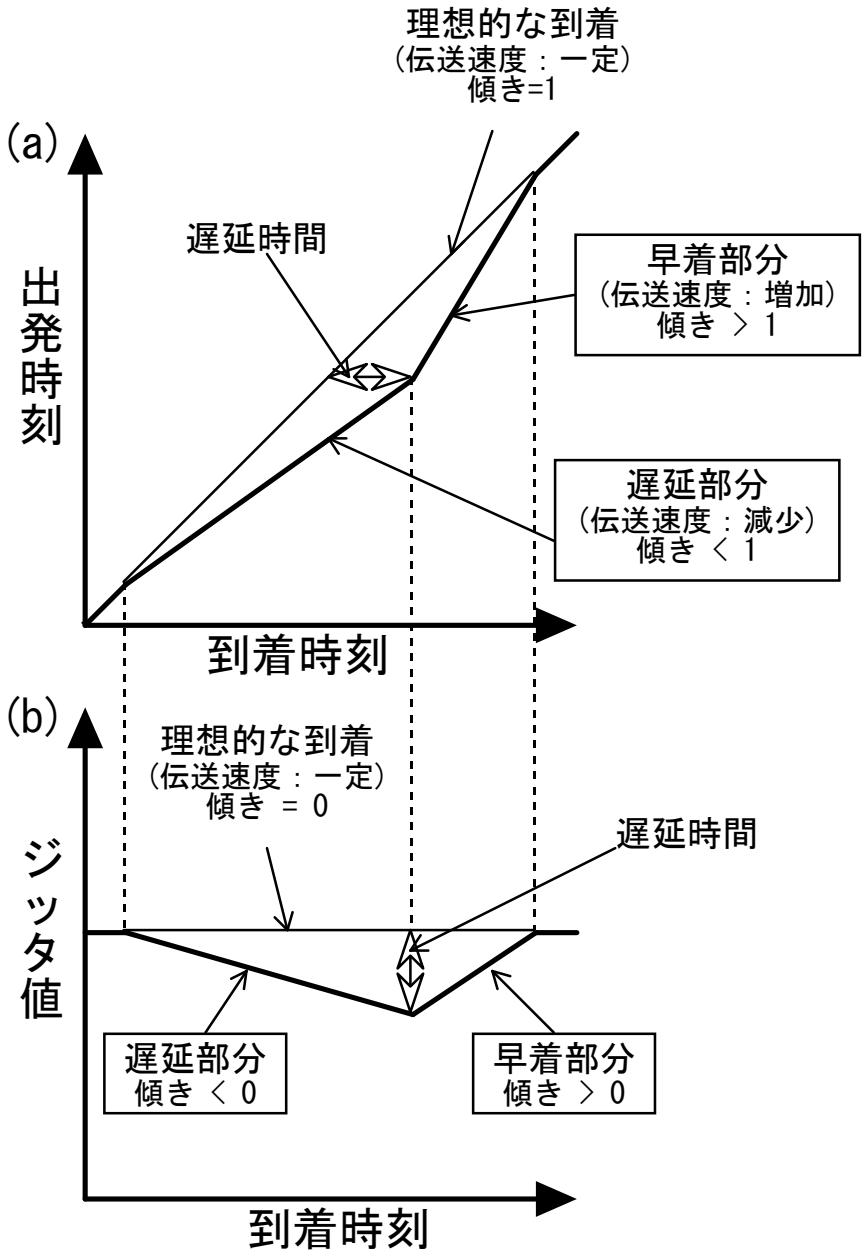


図 5-2-4-3　ジッタの評価方法

図 5-2-4-4 は、測定結果全体を示したものである。全部で 10,000 個のパケットを送出しているので、2500 秒＝41 分 40 秒の測定時間となる。ジッタ値が正の側に振れているのは、パケットが理想的な到着時刻より早く到着していることを示している。測定時間 2,500 秒の間に、±10msec の緩やかなジッタがあることが分かる。ジッタ値が 0 のところ(横軸上)に示されている白丸は、その付近でパケットが欠損したことを示している。パケット自体が消滅しているため、その到着時刻を知ることは出来ないため、最小自乗法で求めた直線の式にパケットの出発時刻を代入し、理想的な到着時刻を求めた。

ジッタ値が小さいところ(正の領域)では、途中の経路上でのバッファに溜まるパケット数が少ないことに対応しているが、この領域でも多くのパケット欠損が発生していることが分かる。例えば、800～900 秒付近では、遅延時間が極小になっているが、この部分で集中的にパケット欠損が発生していることが分かる。ネットワークの混雑が緩和している領域で、多くのパケットが欠損しているため、このメカニズムを調べることが必要である。また図６(a)には、緩やかなジッタ以外に、突発的に遅延時間が増加していることが分かる。これは、他のユーザからのパケットが急激に増加し、遅延時間が増加したと考えられる。しかし、この短期的なジッタとパケット欠損も、対応していないと思われる。表５は、欠損したパケットの番号である。他のパケットを含むため、10,000 以上の番号が存在している。表５から、連続したパケット番号で欠損が発生していないことが分かる。

表 5-2-4-2　欠損したパケットの番号

17	2509	3439	3759	5296	6407	7510	8156	9037	10417
884	2648	3470	4271	5298	6473	7838	8285	9054	10440
1222	2787	3531	4695	5466	6584	7883	8349	9114	
1500	2955	3571	4962	5508	7040	7952	8716	9116	
1977	3105	3654	5076	5703	7227	8063	8788	9311	
2222	3306	3698	5192	5986	7237	8081	8938	9975	
2440	3381	3737	5231	6403	7407	8120	9031	10392	

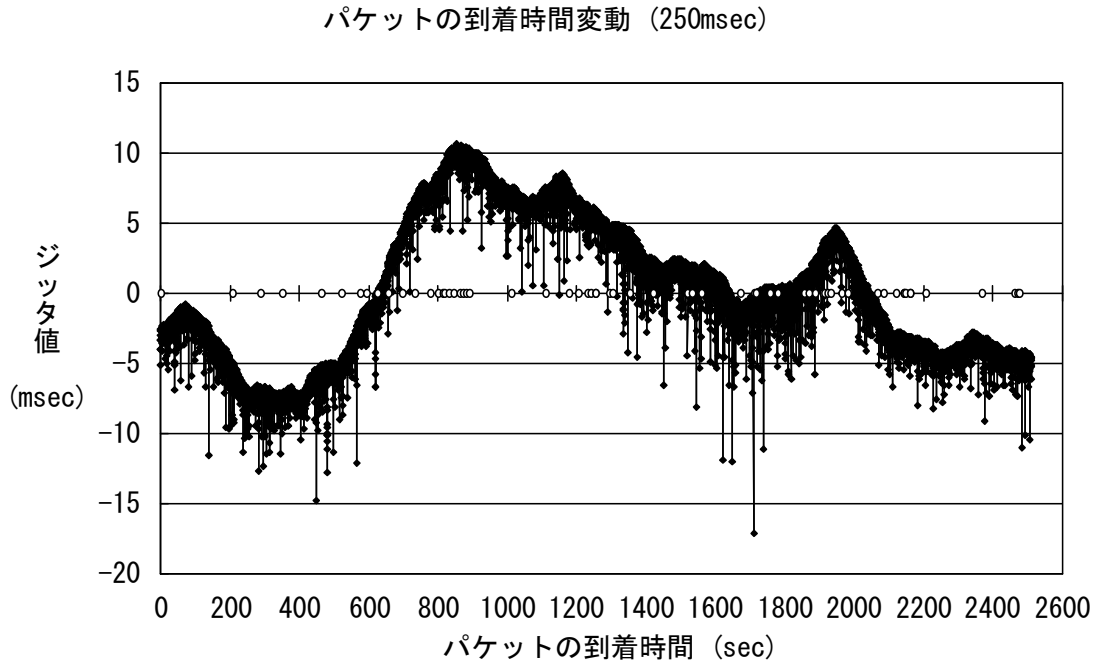


図 5-2-4-4　パケットの到着時刻変動

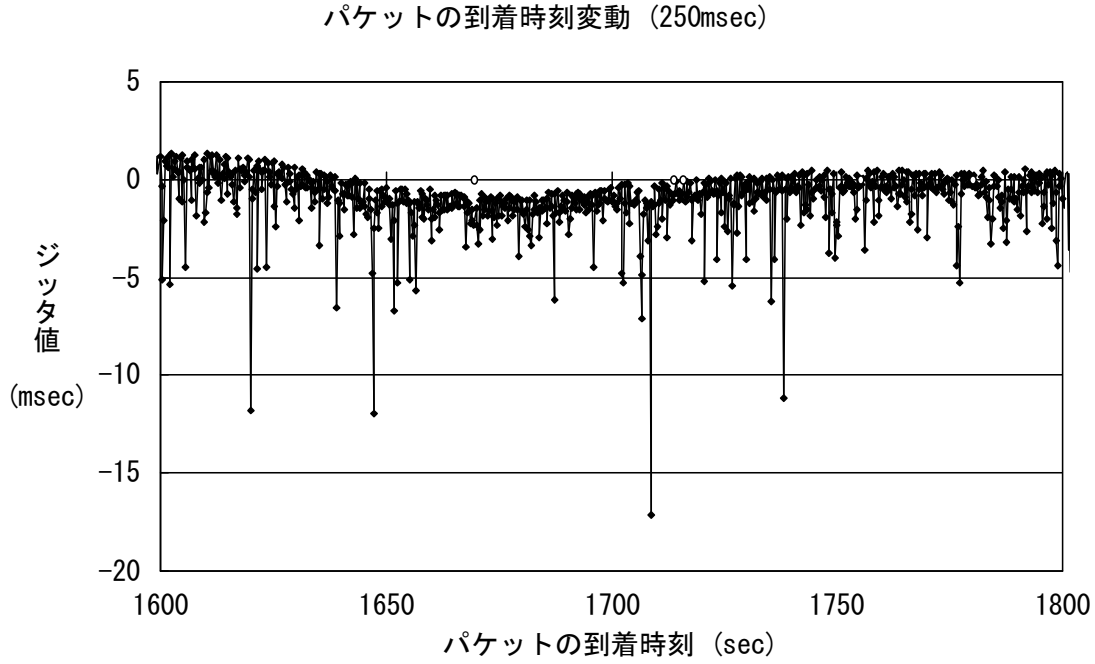


図 5-2-4-5　パケットの到着時刻変動

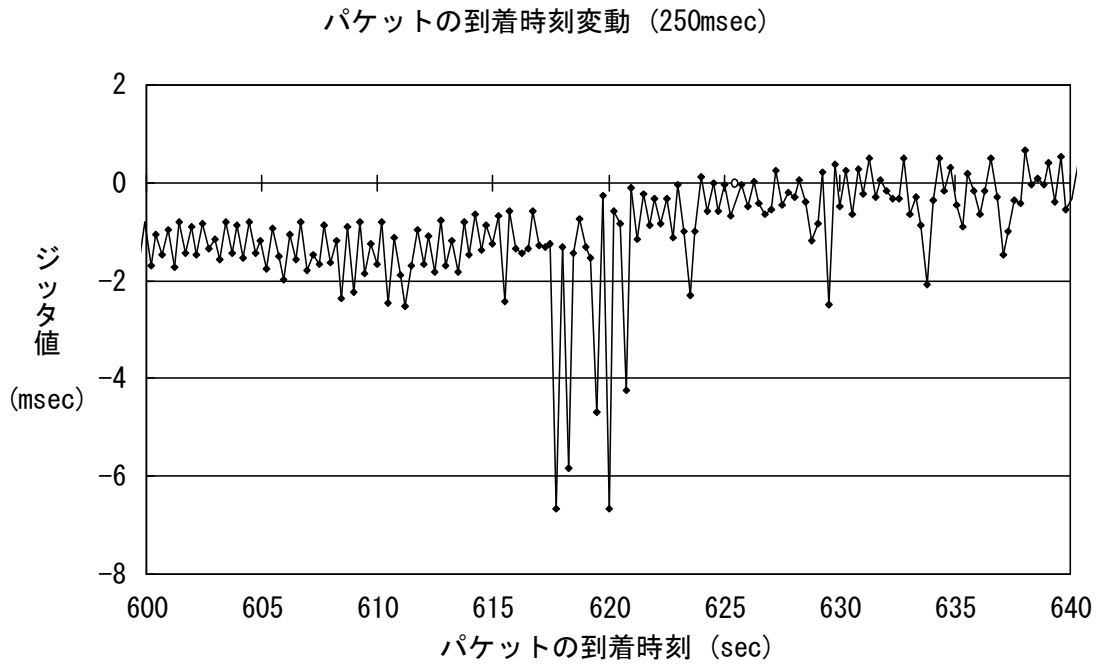


図 5-2-4-6　パケットの到着時刻変動

図 5-2-4-5，5-2-4-6 は、横方向のレンジを拡大したものである。

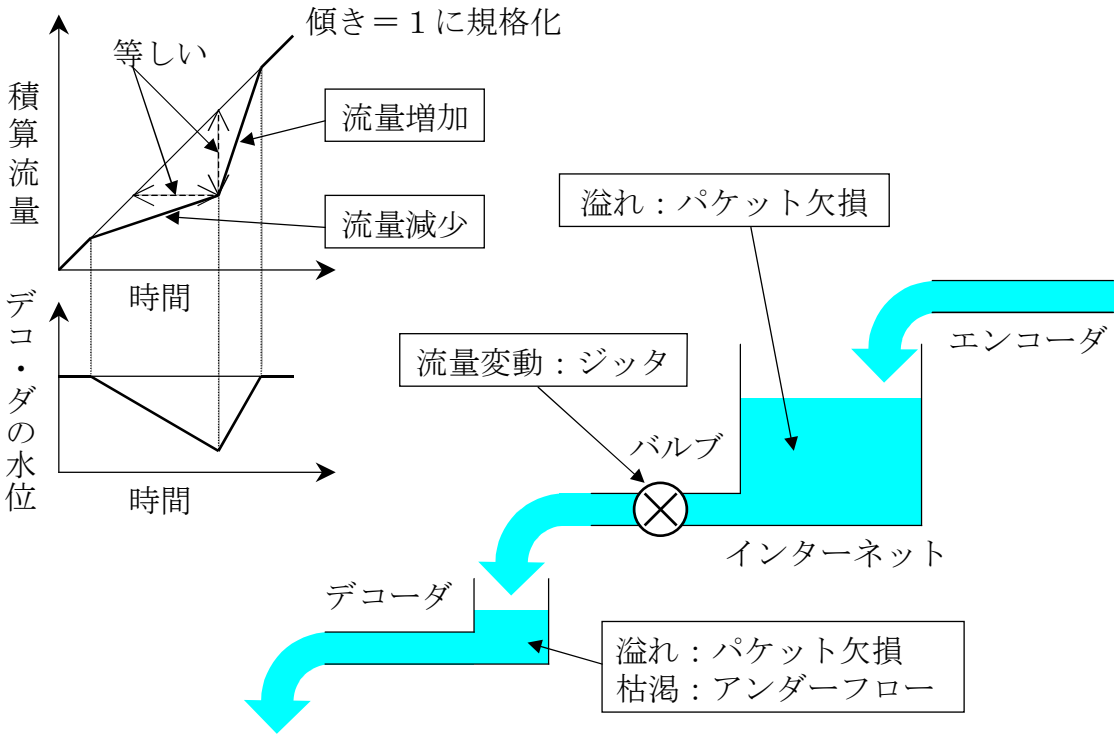


図 5-2-7　インターネットでのジッタ発生モデル

図 5-2-7 は、インターネットでのパケット欠損を模式的にあらわしたものである。エンコーダからは、パケットの流れに相当する水が流れ出し、それをインターネットを模擬したバルブ付きの容器で受け取る。このバルブを絞ったり緩めたりすることが、ジッタに相当する。バルブを緩めることによって、イン

ターネット容器の水量は減少する。一方、バルブを絞ることによって、デコーダに渡す水量は減少し、その影響でインターネット容器の水量が増加する。バルブを絞りすぎると、インターネット容器から水が溢れ、これがパケット欠損に相当する。つまり、インターネット容器からの水量、つまりパケットの時間あたりの送出量をモニタすることによって、インターネット容器の水量を推測することが可能となる。

図 5-2-7 左上に示す様に、時間とエンコーダからの積算水量の関係を規格化して表すと、傾き 1 の直線となる。インターネット容器のバルブを絞るとデコーダに流れ込む水量が減少し、直線の傾きは 1 より小さくなる。過剰に溜まった水を排出するために、バルブを緩めると、インターネット容器から流れ出す水量は増加し、傾きは 1 より大きくなる。エンコーダから流れ出す水量を差し引くと、デコーダ容器の水量を表すグラフが得られる。この直線の傾きが負の場合は、インターネット容器からの水量が減少しており、デコーダ容器の水量が極小値になった際に、インターネット容器の水量が極大になり、インターネット容器から水が溢れやすくなると考えられる。しかし、図 5-2-4-4 の白丸で表されたパケット欠損とジッタの関係は、このモデルとは一致していない。従って、このモデルを再検討する必要がある。

5-2-5　結論

シミュレーションの結果から、ブロックサイズの大きいところで Raptor アルゴリズムは、Reed-Solomon より高速であることが分かった。また、インターネットのジッタには、長い周期のジッタと短い周期のジッタが存在し、緩やかなジッタは測定時間 2,500 秒の間に、±10msec 変動していることが分かった。1%程度のパケット欠損が発生し、この影響によって映像の乱れが発生していると考えられる。

5-3　デバイス化とホームゲートウエイ等評価プラットフォームの試作、実証試験およびＩＥＴＦ等へのドラフト提案

5-3-1　序論

5-2 で述べたように、インターネット上でパケット欠損が発生していることがわかった。FEC（RaptorTM）機能を持たないVODサーバにリアルタイムに符号化できるFECサーバを配備し、復号化機能を有するSTBを用いて、B2C型サービスにおけるFEC技術の有用性・機能性の実証試験を行った。その結果、STB向けVODサービスを対象に商用ネットワーク上でFECの有効性を確認した。

5-3-2　実証実験内容

5-3-2-1　概要

NTT-BBのバックボーンネットワーク内に住友電工ネットワークス製のFECサーバとNEC製の配信サーバを設置し、NTT東日本の地域IP網経由で本サービス用にモニターにSTBを貸与し、FEC技術を適用したMPEG-2 TSデータを配信した。ネットワーク構成は、図5-3-2-1のとおりである。　なお、配信サーバ～STB間は、HSAC1.0(*1)に準拠している。

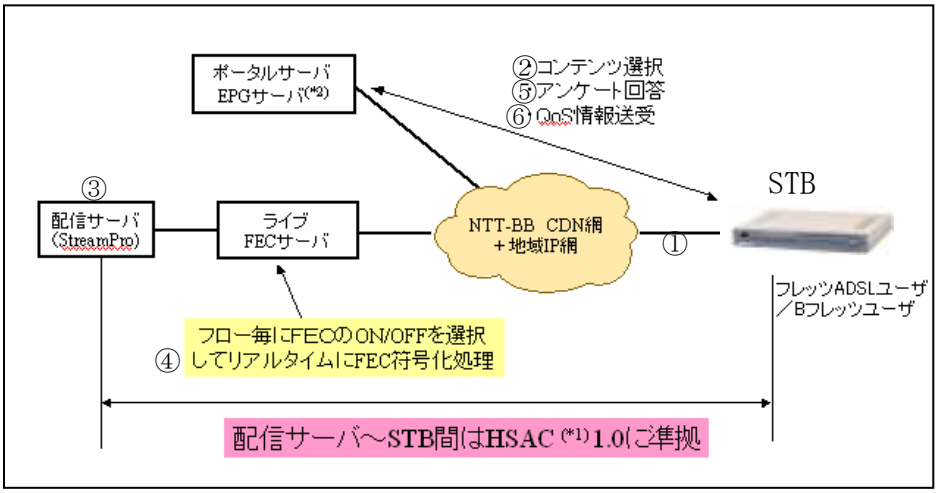


図 5-3-2-1　システム基本構成

(*1)光サービスアーキテクチャコンソーシアム
(*2)Electronic Program Guide（電子番組表）

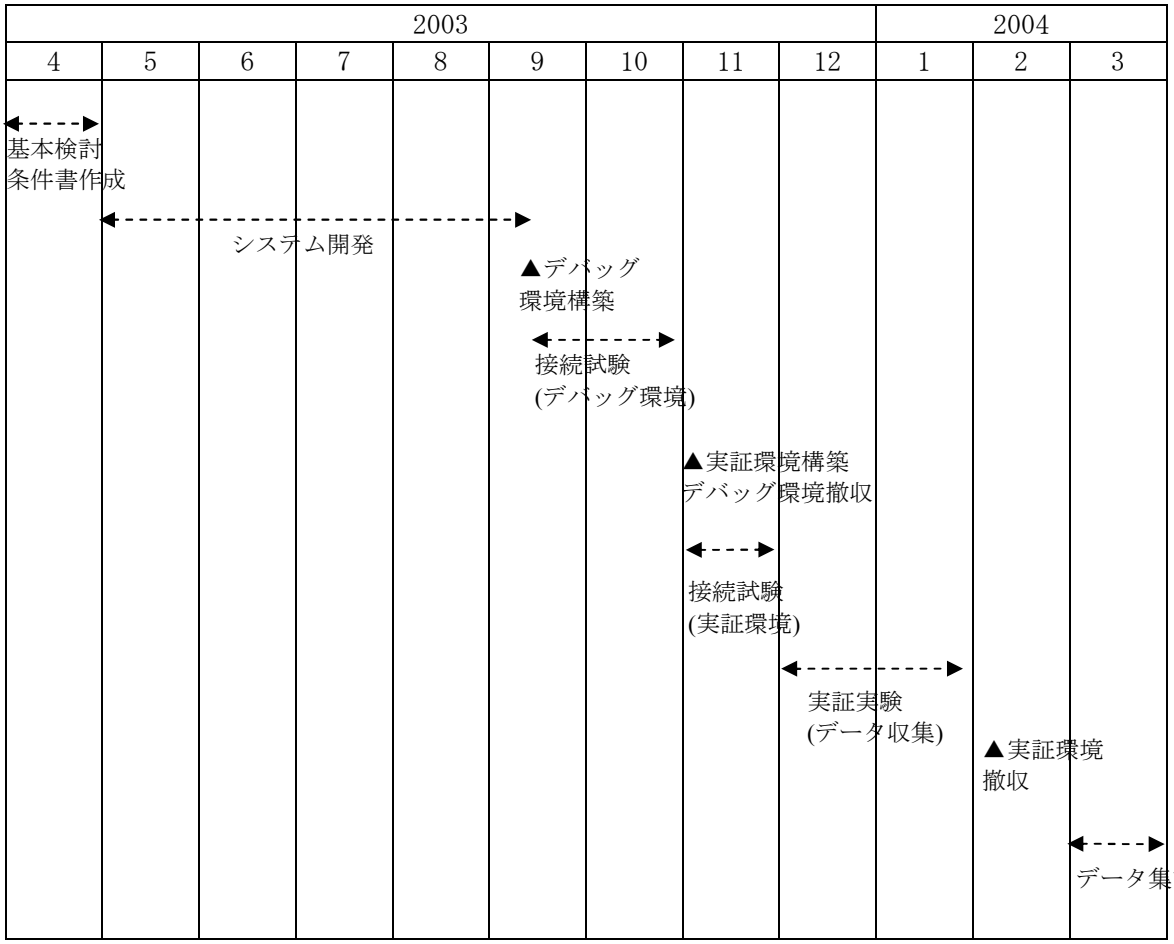
モニターユーザが、次の①～⑥の手順でコンテンツを視聴した後、コンテンツに関する評価を行った。

① フレッツADSLもしくは､Bフレッツを用いてSTBからPPPoEでNTT-BBのBROBA CDN網に接続。
② ポータルサーバにてユーザー認証完了後、STBにポータル(EPG)を表示する。クライアントは、コンテンツを選択。
③ 配信サーバは、STBにMPEG-2 TS形式の映像データを送信。
④ フローごとにFECのON/OFFを選択して、リアルタイムにFEC符号化処理をする。(FEC ON/OFFの切り替えは、ポータルサーバで行う)
⑤ 3分以上の視聴に対して、視聴後、STBにアンケートを表示する。回答されたアンケートは、ポータルサーバに蓄積される。
⑥ QoSデータおよびFEC機能の運用データをFECサーバにsyslogとして送信する。

5-3-2-2 全体工程

実証実験の全体工程表は、表5-3-2-1の通りである。

表5-3-2-1　全体工程表



5-3-2-3 目的

(1)FEC 有無での視聴品質を比較し、その有用性を検証する

以下の観点でユーザー主観評価結果との相関を求める。

・ネットワーク品質

パケットロス率（%）

利用回線種別（フレッツ ADSL、B フレッツ）

ユーザー有効帯域（Mbps）

(2)ネットワーク構成と FEC 適用領域を検証する

各種利用回線のモニターを抽出し、ネットワーク構成別に主観評価を行う。

①利用回線種別

フレッツ ADSL ユーザー：

1. フレッツ・ADSL8M
2. フレッツ・ADSL モア(12M)
3. フレッツ・ADSL モアⅡ(24M)

B フレッツユーザ：

1. ベーシック（100Mbps 占有）
2. ニューファミリー（100Mbps 共用：カプラ使用）
3. マンションタイプ（100Mbps 共用：VDSL 等）

②宅内構成

ブロードバンドルータ有無

宅内帯域の目安（10Mbps/100Mbps）

(3)FEC 有無での操作性を検証する

①映像スタート時の待ち時間への影響

・FEC サーバで設定するバッファ（遅延）によるユーザー操作性向上/劣化の有無検証

②ジャンプ再生時の操作性比較

・操作性の主観評価に違いがでるかを検証

5-3-2-4　システム構成

本システムを構成するサーバの一覧と主な役割は、表 5-3-2-2 のとおりである。

表 5-3-2-2　サーバの一覧と主な役割

設置場所	サーバ名	設置	主な役割
東京拠点	配信管理サーバ	既設	･STBからコンテンツ視聴要求を受付、MPEG配信サーバへ要求をリダイレクションする。 ･MPEG配信サーバから視聴状況を取得し、視聴に関してロギングを行う。 ･MPEG 配信サーバの死活管理・配信許可を行う。
	FEC サーバ	新設	･STBのFEC対応の有無を認識し、ストリーミングデータにFEC情報を付加する。
	スムーザ	新設	･FECサーバから送出されるストリーミングデータをコンテンツビットレートに合わせて平滑化して送出する。L2スイッチとL3スイッチ部で構成される。
	MPEG 配信 サーバ	新設	･配信管理サーバからリダイレクションされた、コンテンツ視聴要求を受け、STBへストリーミングデータを配信する。 配信管理サーバへ視聴状況を報告する。
	ポータル サーバ	新設	･サービス認証を行う。 ･コンテンツ視聴までの、ナビゲーションを行う。 ･コンテンツ視聴後のアンケートの提示、受付、集計を行う。
	ターミナルサーバ	新設	L2SW、L3SW と RS-232 で接続し、リモートコンソールを実現する
センタ	Radius サーバ	既設	･接続認証を行う ･STB へ IP アドレスの払出を行う。
	DAM	既設	･コンテンツの受け入れを行う。 ･コンテンツIDの払出を行う。 ･コンテンツのライフサイクル管理を行う。
	コンテンツ流通 管理サーバ	既設	･DAM で受け入れたコンテンツを MPEG 配信サーバへ流通させる。
	センタ配信管理 サーバ	既設	･コンテンツ管理、ユーザー管理のマスタDBサーバ ･各拠点の配信管理サーバへデータの送出を行う
	LOG 収集サーバ	既設	･(拠点)配信管理サーバから、ログを収集する。

なお、本実験で使用した機器(本実験のために新たに設置した機器)のスペックは表 5-3-2-3 から表 5-3-2-8 のとおりである。

•MPEG 配信サーバ

表 5-3-2-3　MPEG 配信サーバのスペック

機種	NEC Express5800/InternetStreamingServer GS
CPU	Intel Xeon 2.8GHz×1
Memory	1GB
Disk	36GB(15,000rpm)×3(RAID5)
Network	100Base-TX×2(標準 1、増設 1) 1000Base-SX×1(増設 1)
OS	Windows 2000 Server
アプリケーション	StreamPro/Stremaing-MPEG Ver.3.11
外形寸法	(W×D×H) 483×771×88mm
重量	25Kg
消費電力	500W

•配信管理サーバ

表 5-3-2-4　配信管理サーバのスペック

機種	NEC Express5800/InternetStreamingServer GS
CPU	Intel Xeon 2.8GHz×1
Memory	1GB
Disk	36GB(15,000rpm)×3(RAID5)
Network	100Base-TX×2(標準 1、増設 1) 1000Base-SX×1(増設 1)
OS	Windows 2000 Server
アプリケーション	StreamPro/ManagemantSystem Ver3.2
外形寸法	(W×D×H) 483×771×88mm
重量	25Kg
消費電力	500W

•ポータルサーバ

表 5-3-2-5 ポータルサーバのスペック

機種	SUN Fire V120
CPU	Superscalar SPARC Version 9 UltraSPARC IIi 650MHz
Memory	1GB
Disk	36.4GB UltraSCSI ×2 台 (10000rpm)
Network	10BASE-T/100BASE-TX ×2
OS	Solaris8
アプリケーション	ftpd, telnet, OracleDB, Oracle Application Server, StreamingView
外形寸法	(W×D×H) 436.7×478×43.6mm
重量	10Kg
消費電力	150W

•FEC サーバ

表 5-3-2-6 FEC サーバのスペック

機種	HP Proliant DL360
CPU	Intel Xeon 2.8GHz
Memory	2Gbps
Disk	36.4GB（10,000rpm）
Network	10/100/1000BASE-TX×2 ポート（オンボード） 1000BASE-SX×2 ポート（追加）
OS	RedHat8.0
アプリケーション	vsftpd、telnetd、livefecd(住友電工)
FEC パラメータ	/home/taofec/bws/livefecd.conf に下記のパラメータを設定 Raptor のプロトコル番号：High Performance Mode Source Block Size：64KB Encoding Unit Size：1024Byte Packet Loss Rate(%)：1.0% Target Loss Rate(%)：1e-9
外形寸法	436(W)x692(D)x43(H) mm
重量	15Kg
消費電力	325w

•スムーザ (L2SW、 L3SW)

表 5-3-2-7 スムーザのスペック

機種	NEC IP8800 / 710B
CPU	-
Memory	-
Disk	-
Network	100base-TX×16、1000base-SX×1
OS	-
アプリケーション	-
外形寸法	(W×D×H)435×470×88mm
重量	9.8Kg
消費電力	150W

•ターミナルサーバ

表 5-3-2-8 ターミナルサーバのスペック

機種	MOXA Nport DE-302
CPU	80186
Memory	512KB
Disk	-
Network	10/100Base-TX×1
OS	-
アプリケーション	-
外形寸法	(W×D×H)155×105×33 mm
重量	9.8Kg
消費電力	9～30V DC, 1.05A/9V

5-3-2-5　シーケンス図

再生開始までの RTSP プロトコルのシーケンス図は、図 5-3-2-2 の通りである。クライアントが送信する RTSP SETUP メソッドに FEC 用拡張ヘッダを付けて FEC 配信要求を行う。FEC サーバは、RTSP SETUP メソッドに FEC 拡張ヘッダが付いているセッションの RTP データに FEC 処理を行う。なお、この FEC 付与のプロセスに関しては、IETF にて標準化活動を行う予定である。

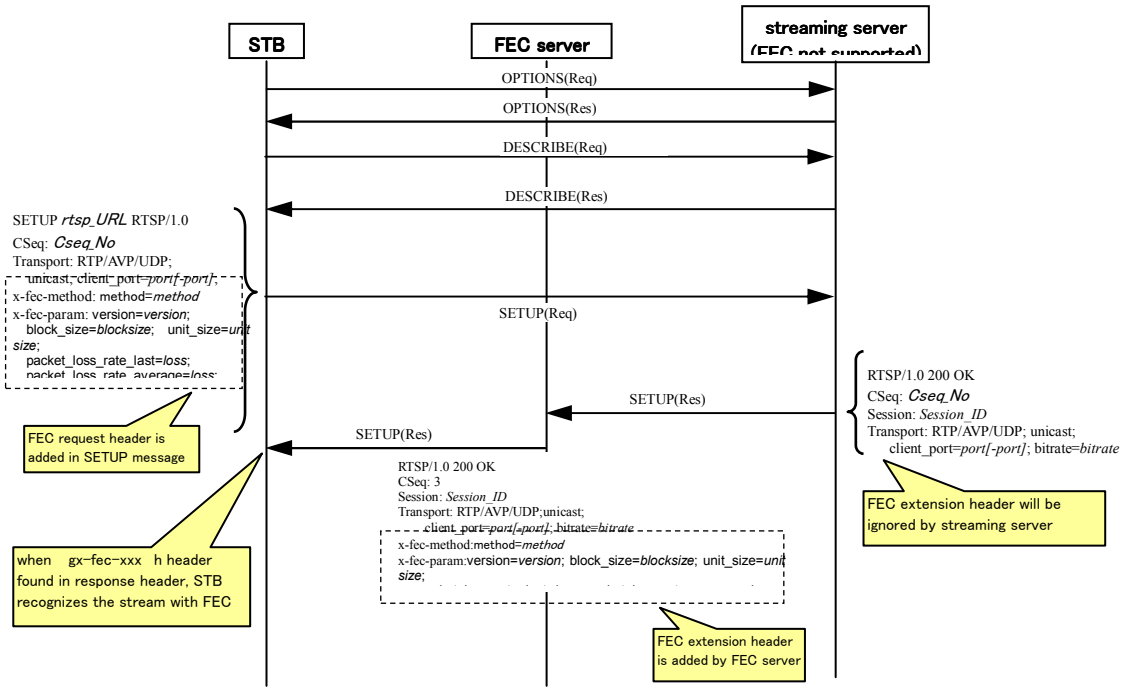


図 5-3-2-2　RTSP プロトコルのシーケンス図

5-3-2-6　実験期間

モニター期間は、平成 15 年 12 月 1 日～平成 16 年 1 月 31 日の 2 ヶ月間であった。ただし、12 月 1 日からモニター機器を順次発送したため、収集データは、12 月 8 日からのものを採用した。

5-3-2-7　モニターの内訳

モニターは、東京都（一部エリアを除く）において、NTT 東日本の提供する「フレッツ・ADSL（モアⅡ、モア、8M タイプ）」、「B フレッツ（ベーシック、ニューファミリー、ファミリー、マンション）」を利用し、実効速度が 4Mbps 以上の利用環境にあるユーザーに限定して募集した。

モニターは、都内在住の合計 299 人を対象とする。

年齢、性別、利用回線種別の内訳は、次の通りである。

〈年代内訳〉

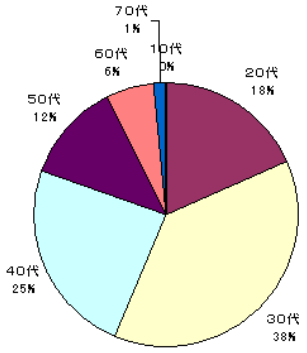
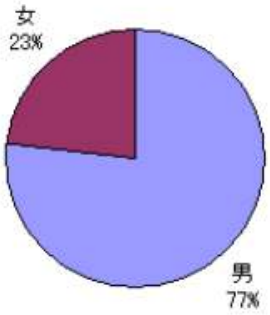


図 5-3-2-3　年代内訳

	人数	構成比
10	1人	0%
20	55人	18%
30	112人	37%
40	72人	24%
50	37人	12%
60	18人	6%
70	4人	1%
	299人	100%

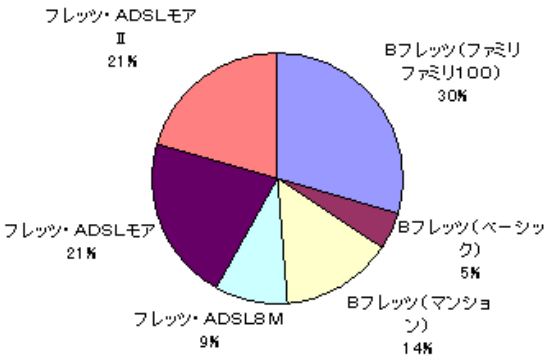
〈性別〉



	人数	構成比
男	229人	77%
女	70人	23%

図 5-3-2-4　性別内訳

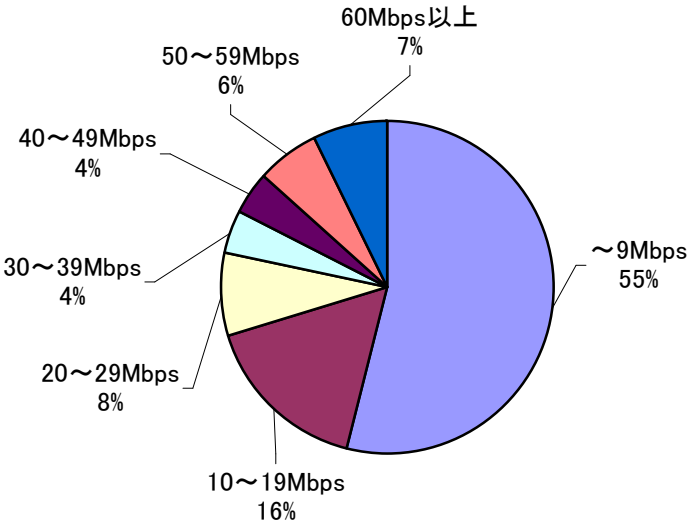
〈利用回線種別内訳〉



	人数	構成比	
Bフレッツファミリ	89人	30%	Bフレッツ(光)計48%
Bフレッツベーシック	14人	5%	
Bフレッツマンション	42人	14%	
フレッツ・ADSL8M	28人	9%	フレッツADSL計52%
フレッツ・ADSLモア	64人	21%	
フレッツ・ADSLモアⅡ	62人	21%	

図 5-3-2-5 利用回線種別内訳

〈モニターの実効速度〉



実行速度	人　数	構成比
～9Mbps	161人	55%
10～19Mbps	49人	16%
20～29Mbps	24人	8%
30～39Mbps	13人	4%
40～49Mbps	12人	4%
50～59Mbps	18人	6%
60Mbps以上	22人	7%

図 5-3-2-6 モニターの実効速度

〈利用形態〉

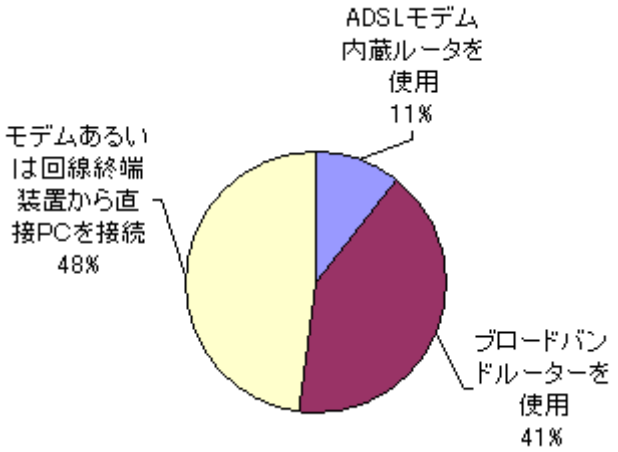


図 5-3-2-7　利用形態

接続形態	人数	構成比
ADSLモデム内蔵ルータを使用	32人	11%
ブロードバンドルーターを使用	123人	10%
モデムあるいは回線終端装置から直接PCを接続	144人	48%

5-3-2-8　実験に使用したコンテンツ

「アニメ」、「映画」、「音楽」の大きく分けて 3 種類、約 100 本のコンテンツを提供した。コンテンツの中には、FEC-ON・FEC-OFF が混在する。
なお、コンテンツは、表 5-3-2-9 の通りにエンコードしたものを使用した。

表 5-3-2-9　エンコード条件

エンコーダボード	MPEG MovieMaKer 200S Publisher / Networker / Basic
フォーマット形式	MPEG-2 TS
ビットレート形式	CBR
ビットレート	3Mbps、6Mbps
画面サイズ	720×576 以下
フレームレート	29.97fps (NTSC)
GOP 構成	N=15 以下、M=3 以下
Video Elementary Stream	1)Video bit streamSequence header から開始 2) 開始は I ピクチャ、かつ直前は GOP Header を含む
Audio Elementary Stream	1)Audio bit srearmLayerⅡ : AAU Header から開始 2)ビットレートインデックスがフリーフォーマットは不可
その他	・PAT,PMT が定期的に含まれていることが必要(0.5 秒に一回程度) ・Video Elementary Stream,Audio Elementary Stream がそれぞれ 1 ストリームであること ・「Video なし」あるいは「Audio なし」は不可

5-3-2-9　評価方法

実験には、定量的(客観)評価と定性的(主観)評価の 2 つの評価方法を用いた。

(1)定量的(客観)評価

STB に QoS 評価用エージェントを実装し、コンテンツ視聴毎に QoS データおよび FEC 機能の運用データを収集した。QoS 評価用サーバに対してサーバ・クライアント間の QoS 情報及び FEC 機能の運用情報を通知する。

【評価項目】

・パケットロス
・FEC 修復失敗エラー
・FEC 適用ステータス

(2)定性的(主観)評価

定性的評価では、視聴毎に STB による簡易なアンケートと、Web による詳細なアンケートを採った。

〈STB によるアンケート〉

コンテンツ視聴毎に下記の評価項目に関するアンケート画面を表示するようにして STB 付属のリモコンで 3 段階（よい、ふつう、わるい）の選択回答方式とした。この時、視聴しているコンテンツに FEC が適用されているかどうかはモニターに告知しない。

【評価項目】

・映像品質
・音声品質
・操作性
・応答性
・デザイン・視認性
・コンテンツ良否

〈Web 上でのアンケート〉

視聴画面でのアンケート集計を補完するため、視聴品質について Web アンケートに PC で回答してもらった。

5-3-2-10　収集データ

定量的評価の元となるデータは、STB・LNS・配信管理サーバ・ポータルサーバ・LOG 収集サーバの計 5 箇所で収集する。各箇所で収集可能なデータは、表 5-3-2-10 の通りである。

表 5-3-2-10　各データ収集ポイント（5 箇所）で収集可能なデータ一覧

No.	データ収集ポイント	収集可能項目	収集頻度	データベースへの登録	収集データの役割
1	STB	・受信パケット数 ・欠損パケット数 ・パケットロス率 ・FECデコード失敗率 ・FEC適用ステータス ・エラー番号	視聴毎	視聴毎	・ユーザID特定 ・STB　ID特定 ・NW品質特定
2	LNS	モニタが属するLNSポートのトラフィック（上り、下り）（但し、モニタ以外のトラフィックも含めた当該LNSポートのトラフィック合計）	1日単位	1日単位	NW品質特定（補助）
3	配信管理サーバ	・クライアントIPアドレス ・ストリーミングサーバIPアドレス ・サーバタイプ（今回MPEG） ・コンテンツURL ・ユーザID（今回1つに統一） ・視聴開始日付、時間 ・視聴終了日付、時間 ・再生時間 ・配信バイト数 ・終了コード ・メディア/プロトコル情報	視聴毎	1回/1日	・STB特定 ・コンテンツ特定 ・視聴時間特定 ・STBとの通信エラー有無特定
4	ポータルサーバ	・クライアントIPアドレス ・ストリーミングサーバIPアドレス ・コンテンツURL ・サービス認証ID ・視聴開始日付、時間 ・視聴終了日付、時間 ・終了コード ・アンケート結果 ・映像品質/音声品質/操作性/応答性 ・デザイン・視認性/コンテンツ良否	視聴毎	アンケート回収毎	・主観評価収集 ・サービス認証IDとIP特定 ・コンテンツ特定 ・視聴時間特定 ・STBとの通信エラー有無特定
5	LOG収集サーバ	配信管理サーバに同じ	1回/1日	1回/1日	

5-3-2-11　アンケート内容

定性的評価の元となるアンケートは、STB と Web(PC)で採取する。

(1) STB で採取するアンケートの内容

STBで採取するアンケートは、期間ごとに次のような質問内容で行う。コンテンツ視聴後、よい・ふつう・わるい の3段階の選択回答方式とする。

- 平成 15 年 12 月 8 日～平成 16 年 1 月 8 日のアンケート内容

コンテンツの内容はいかがでしたか
最後まで視聴できましたか
コンテンツに興味はありましたか
視聴は安定していましたか
画質はよかったですか
映像の動きはよかったですか
音質はよかったですか
早送りなどの応答性はよいですか
STB の操作性はよいですか
パソコンでの視聴と比べて満足ですか
無料コンテンツとしたら満足ですか

- 平成 16 年 1 月 9 日～平成 16 年 1 月 31 日のアンケート内容

コンテンツの品質に関するアンケートに絞った。

コンテンツの内容はいかがでしたか
最後まで視聴できましたか
視聴は安定していましたか
画質はよかったですか
映像の動きはよかったですか
音質はよかったですか

(2) Web で採取するアンケートの内容

- コンテンツ 3M、6M 品質の比較について
  - ・視聴成功の割合
  - ・視聴の際の安定感
  - ・画質の比較
- 画質について
  - ・画質の満足度
- 音質について
  - ・音質の満足度
- エラー、不具合など視聴した結果発生した事象について
  - ・意図しない中断発生の割合
  - ・不具合の事象

5-3-3　実験結果

5-3-3-1　日別のアクセス傾向

日別のアクセス傾向は、図5-3-3-1の通りである。土日は、ほぼ毎週100以上のアクセスがあった。これは、平日より時間に余裕があり、じっくりとコンテンツ視聴ができる環境にあると思われる。また、新たなコンテンツが視聴開始になる月曜日にもアクセスが多い。週のなかば(特に水・木曜日)は、アクセスが落ちる傾向にある。最終日(1/31)の総アクセス数が一番多かった。

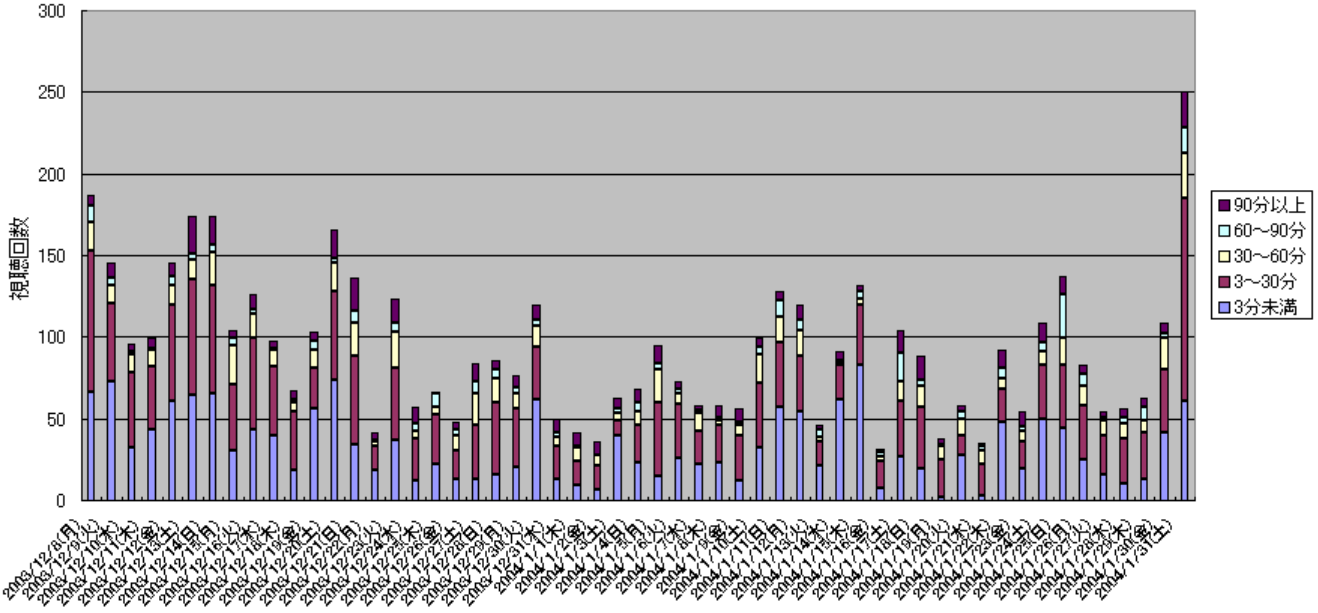


図5-3-3-1　総アクセス数(2003年12月8日～2004年1月31日)

総アクセス数のうち、視聴時間が「3分未満」、「3～30分」、「30～60分」、「60～90分」、「90分以上」の5つに分類して集計したところ、3～30分が最も多く、次いで3分未満が多かった。また、60分以上視聴している割合は、14%程度であった。

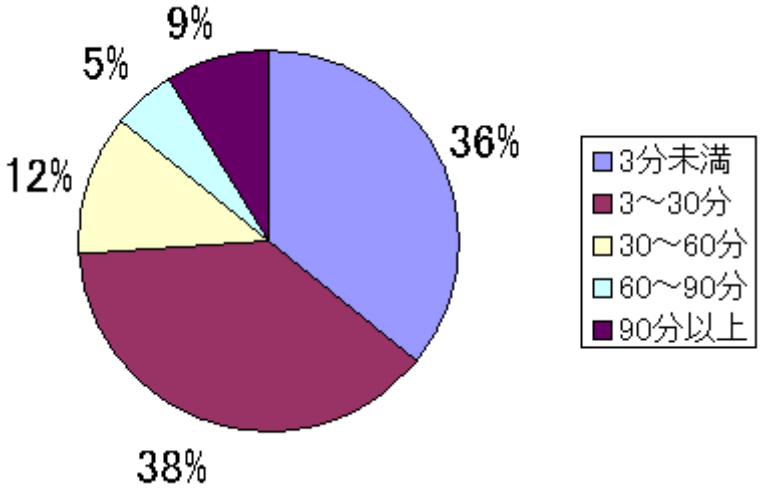


図5-3-3-2　視聴時間内訳

サーバへの同時アクセス数については、昼間より夜間の方が多い傾向にある。また、総アクセス数と同様、週末にアクセスが多い。同時アクセス数のピーク数は、14クライアント(全体の約5%)である。年末年始のアクセス傾向については、週末と同程度のアクセスがあると想定していたが、意外に少なかった。

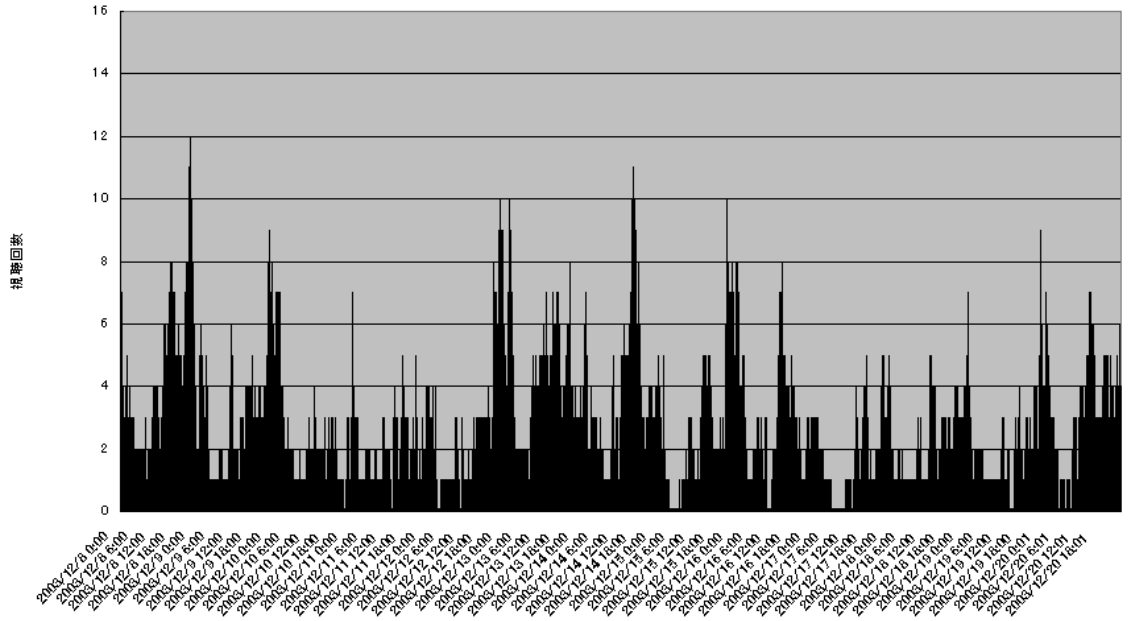


図 5-3-3-3　同時アクセス数(2003 年 12 月 8 日～2003 年 12 月 20 日)

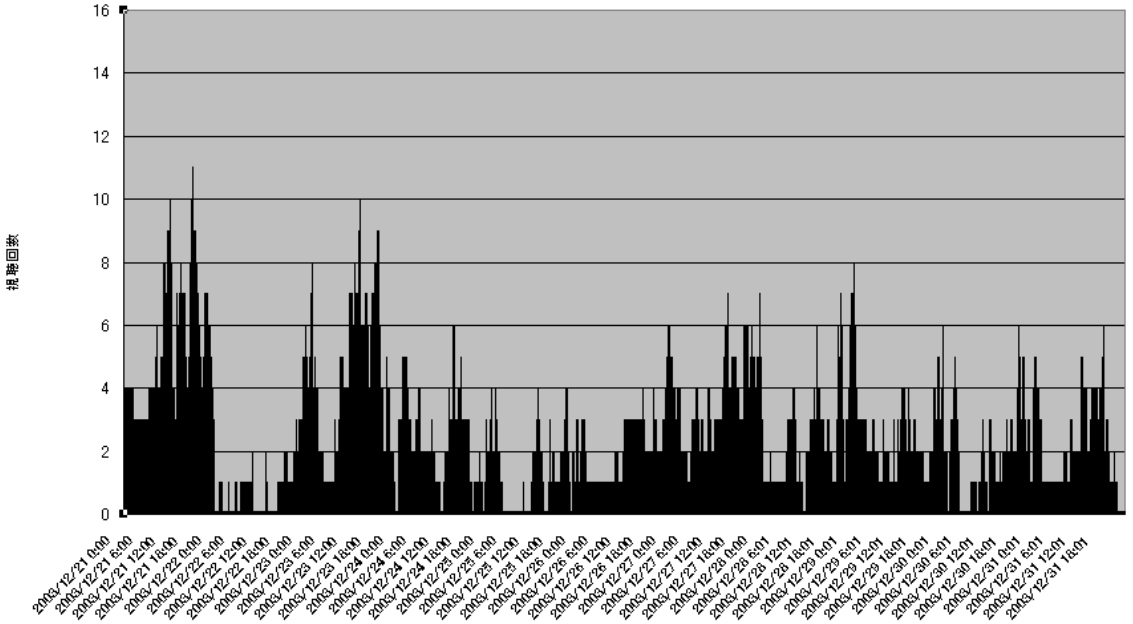


図 5-3-3-4　同時アクセス数(2003 年 12 月 21 日～2003 年 12 月 31 日)

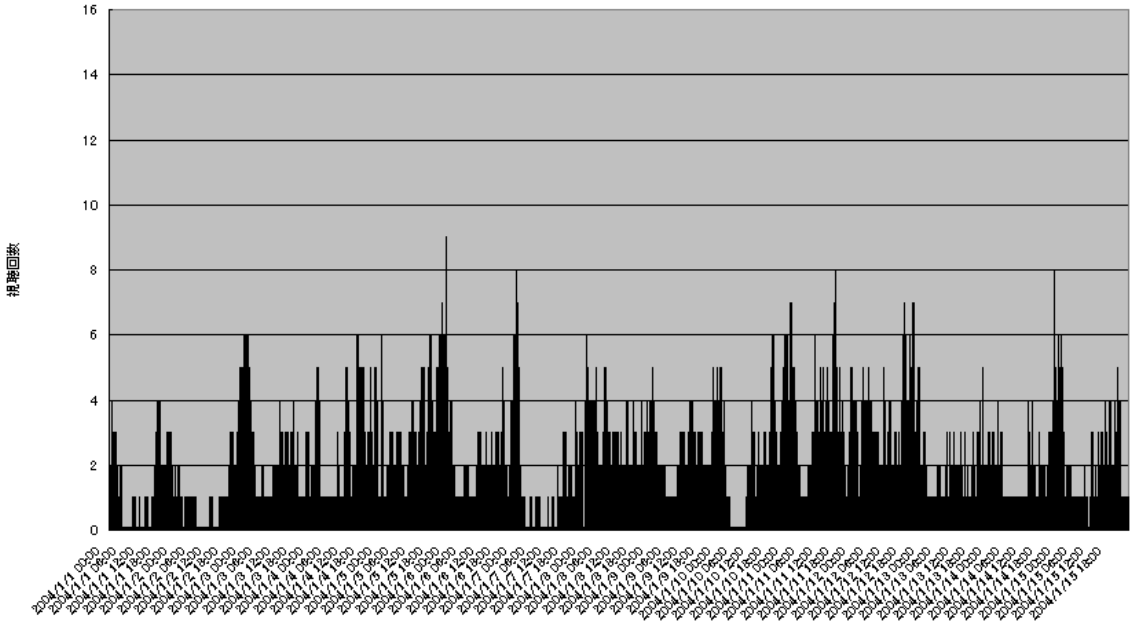


図 5-3-3-5　同時アクセス数(2004 年 1 月 1 日～2004 年 1 月 15 日)

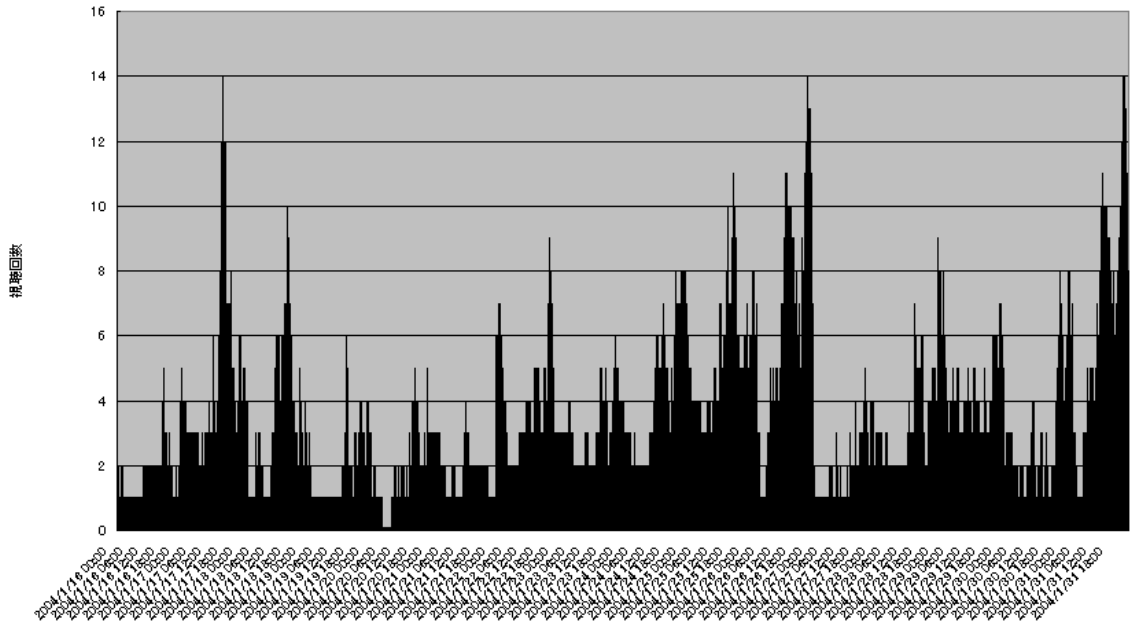


図 5-3-3-6　同時アクセス数(2004 年 1 月 15 日～2004 年 1 月 31 日)

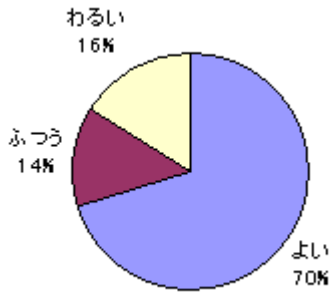
5-3-3-2　FEC 有無の比較

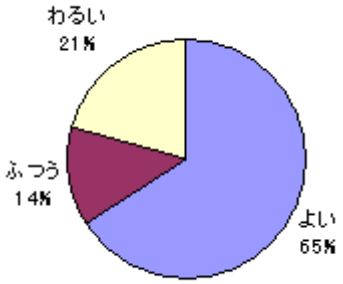
FEC の有用性を検証するために、FEC の有無によってどのような違いがあるかどうか、アンケート結果や実験データ(視聴時間、パケットロス率など)をもとに比較した。

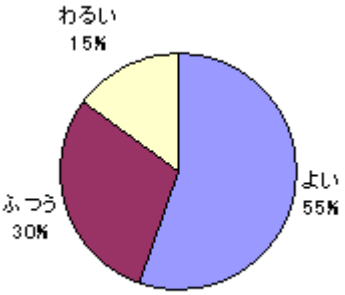
(1) アンケート結果を比較

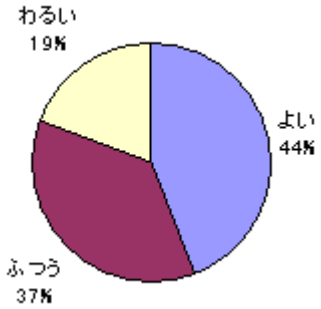
モニターには FEC の有無を告知せずに、アンケートを行ったところ、すべてのアンケートにおいて、FEC-ON の方が優れているという回答結果であった。特に、FEC-OFF の場合に「わるい」という評価をつけるモニターが多かった。

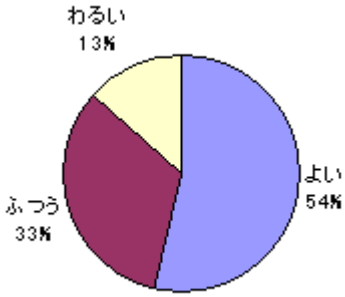
・コンテンツの品質に関するアンケート
(平成 15 年 12 月 8 日～平成 16 年 1 月 31 日)

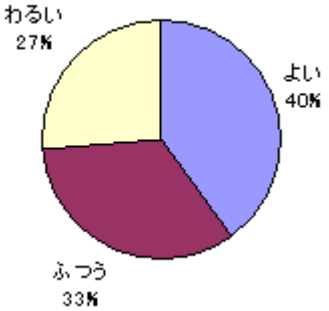
<table>
<tr><td colspan="2">最後まで視聴できましたか</td></tr>
<tr><td> 

FEC-ON</td><td> 

FEC-OFF</td></tr>
<tr><td colspan="2">視聴は安定していましたか</td></tr>
<tr><td> 

FEC-ON</td><td> 

FEC-OFF</td></tr>
<tr><td colspan="2">画質はよかったですか</td></tr>
<tr><td> 

FEC-ON</td><td> 

FEC-OFF</td></tr>
<tr><td colspan="2">映像の動きはよかったですか</td></tr>
</table>

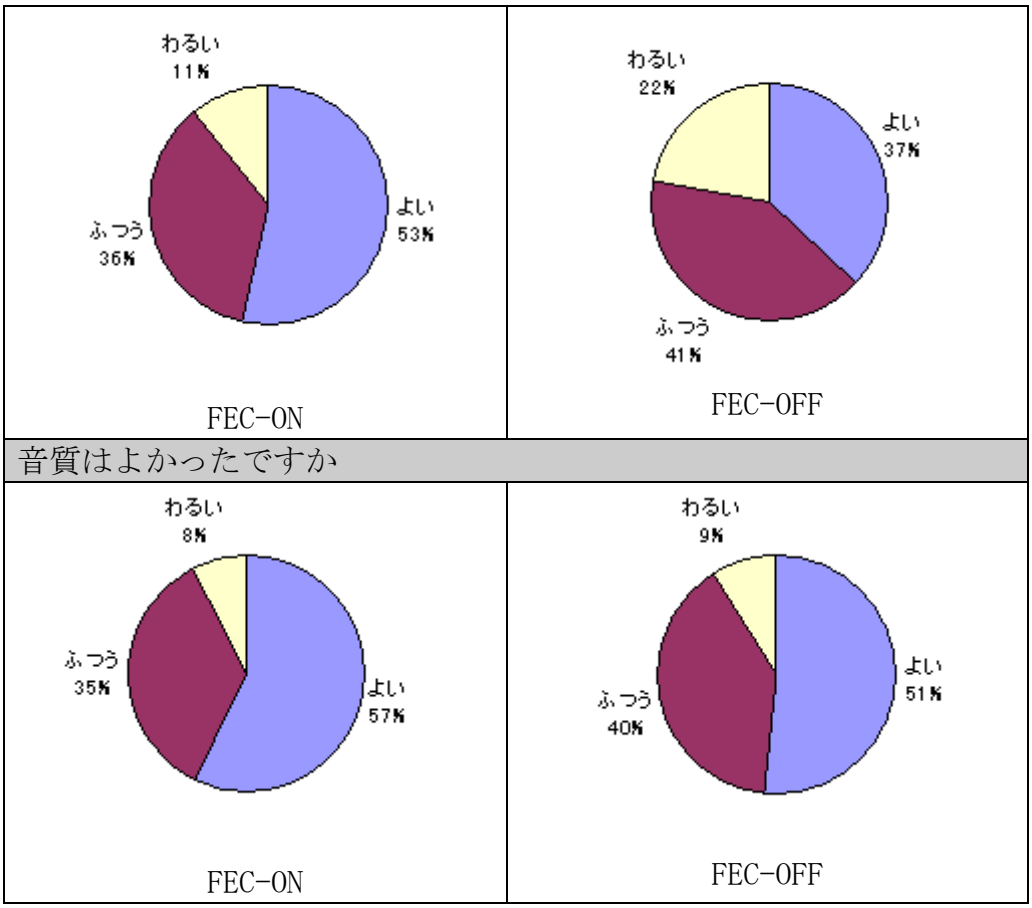
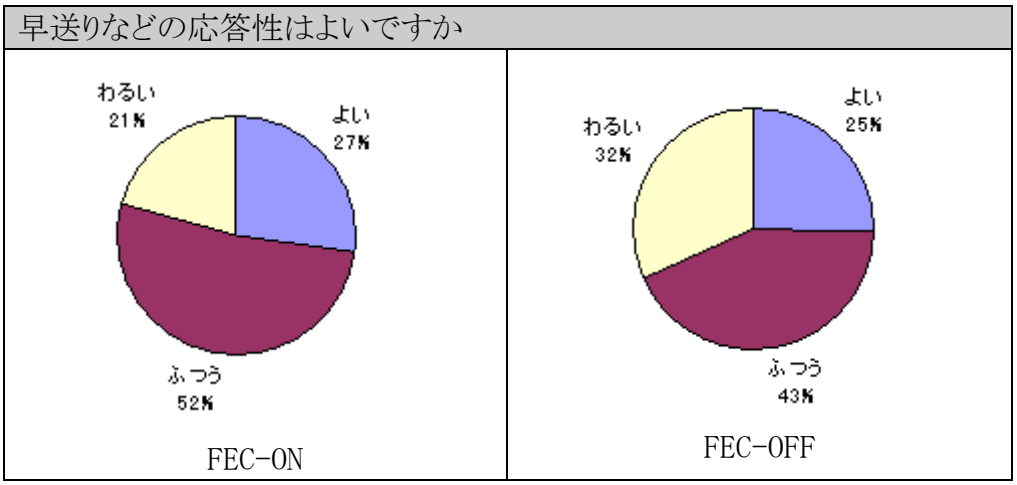


図 5-3-3-7　コンテンツの品質に関するアンケート

・操作性に関するアンケート
(平成 15 年 12 月 8 日～平成 16 年 1 月 8 日)

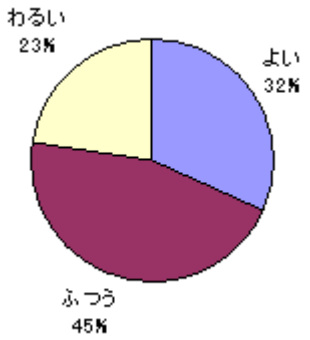
FEC-ON の操作性・応答性に関しては、30%近くのモニターが「よい」と答え、「ふつう」も含めると約 80%を占めている。パソコンとの比較については、約半数のモニターが、パソコンでの視聴と比べて満足していると答えていることが分かる。

また、FEC を付与することによって、トリックプレイ(早送りや巻き戻し)の操作性が向上する傾向にある。



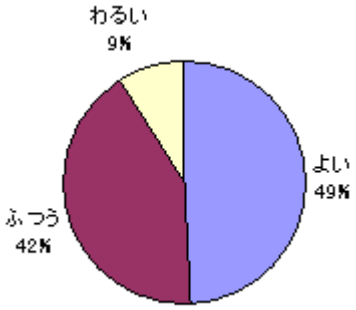
STB の操作性はよいですか

FEC-ON

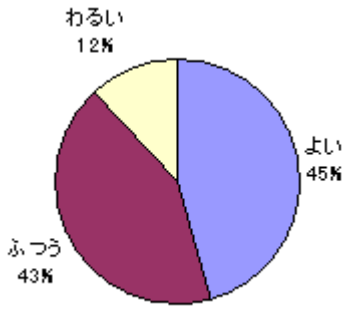


FEC-OFF

パソコンでの視聴と比べて満足ですか



FEC-ON



FEC-OFF

図 5-3-3-8　操作性に関するアンケート

(2) パケットロス率と視聴時間の関係を比較

モニターのパケットロス率と視聴時間の関係をグラフにしてみると、FEC-ON の場合(図 5-3-3-9)には、パケットロスが 10%以下であれば、多くのモニターが 20 分以上視聴しているのが分かる。一方、FEC-OFF の場合(図 5-3-3-10)は、パケットロスが 1%以下でも、視聴に耐えられず、ほとんどのモニターが視聴を止めてしまうことが分かる。つまり、FEC の効果によって、モニターの視聴時間が長くなる傾向にある。

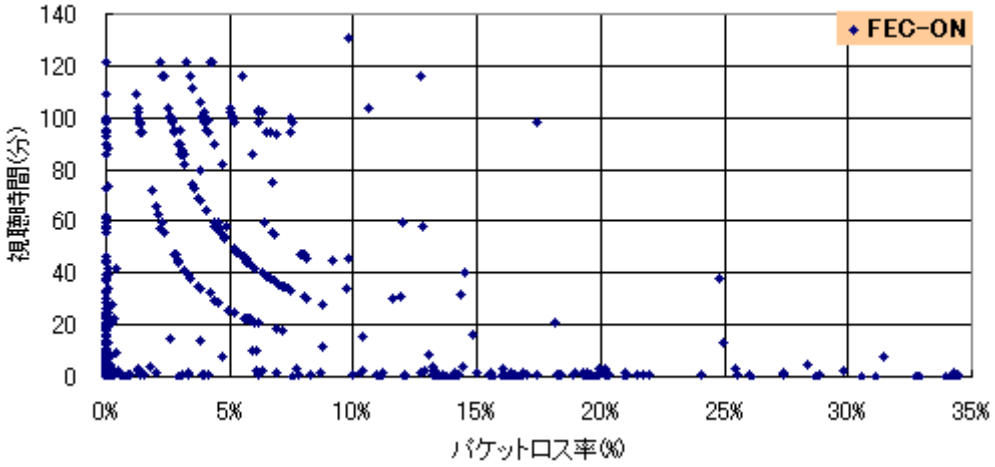


図 5-3-3-9 モニターのパケットロス率と視聴時間の関係(FEC-ON)

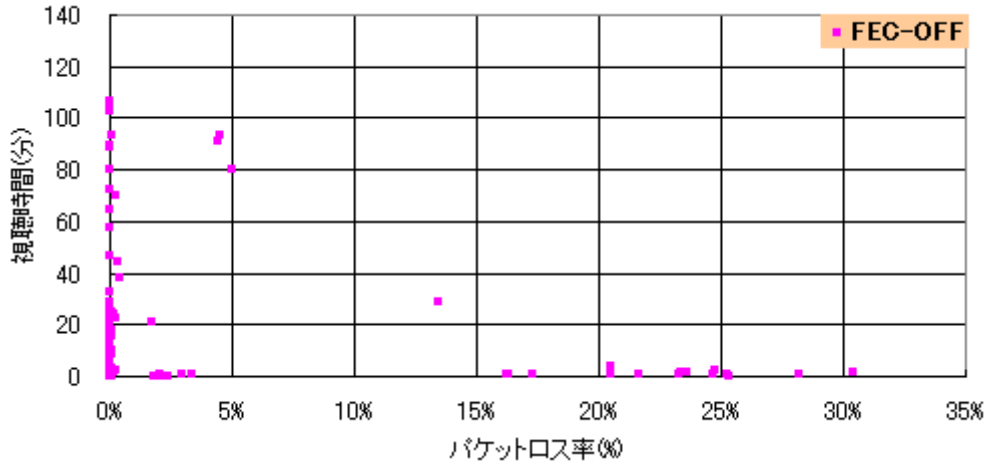
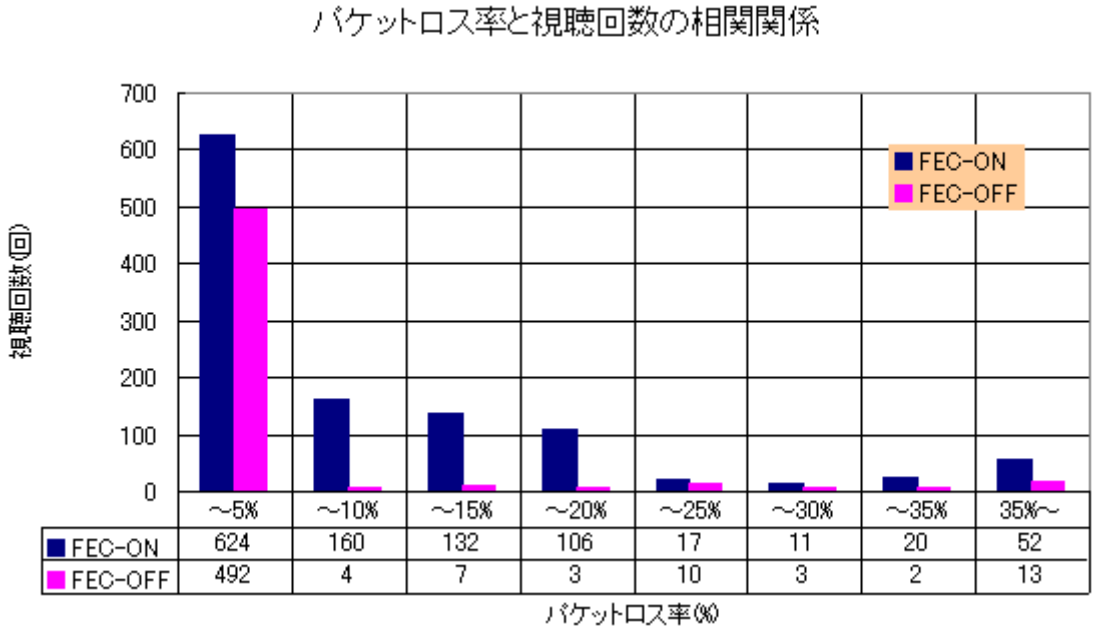


図 5-3-3-10 モニターのパケットロス率と視聴時間の関係(FEC-OFF)

(3) パケットロス率と視聴回数の関係を比較

各モニターのパケットロス率と(3 分以上の)視聴回数の関係を図 5-3-3-11 に示す。FEC-ON の場合には、パケットロスが 20%以下でも 100 以上の視聴回数があるのに対して、FEC-OFF の場合は、パケットロスが 5%を超えてしまう(グラフ上は5%刻みで表記しているが実際は1%以下)と、ほとんどアクセスがなくなってしまう。つまり、FEC の効果によって、モニターの視聴回数が増える傾向にある。



	～5%	～10%	～15%	～20%	～25%	～30%	～35%	35%～
FEC-ON	624	160	132	106	17	11	20	52
FEC-OFF	492	4	7	3	10	3	2	13

図 5-3-3-11 パケットロス率と視聴回数の関係

(4) パケットロス率と安定度評価の関係を比較

安定度が「よい」と答えたモニターのパケットロス率の内訳を算出したところ、FEC-ON では、パケットロス 5%までが 65%、5～10%が 33%であった。一方、FEC-OFF では、パケットロス 5%まで（実際の測定値は1％以下）が 99%であった。つまり、FEC-ON では、約 10%までのパケットロス発生しても安定したサービスが提供できるが、FEC-OFF では、パケットロスが1％以下でも、サービス提供が困難となってしまう。

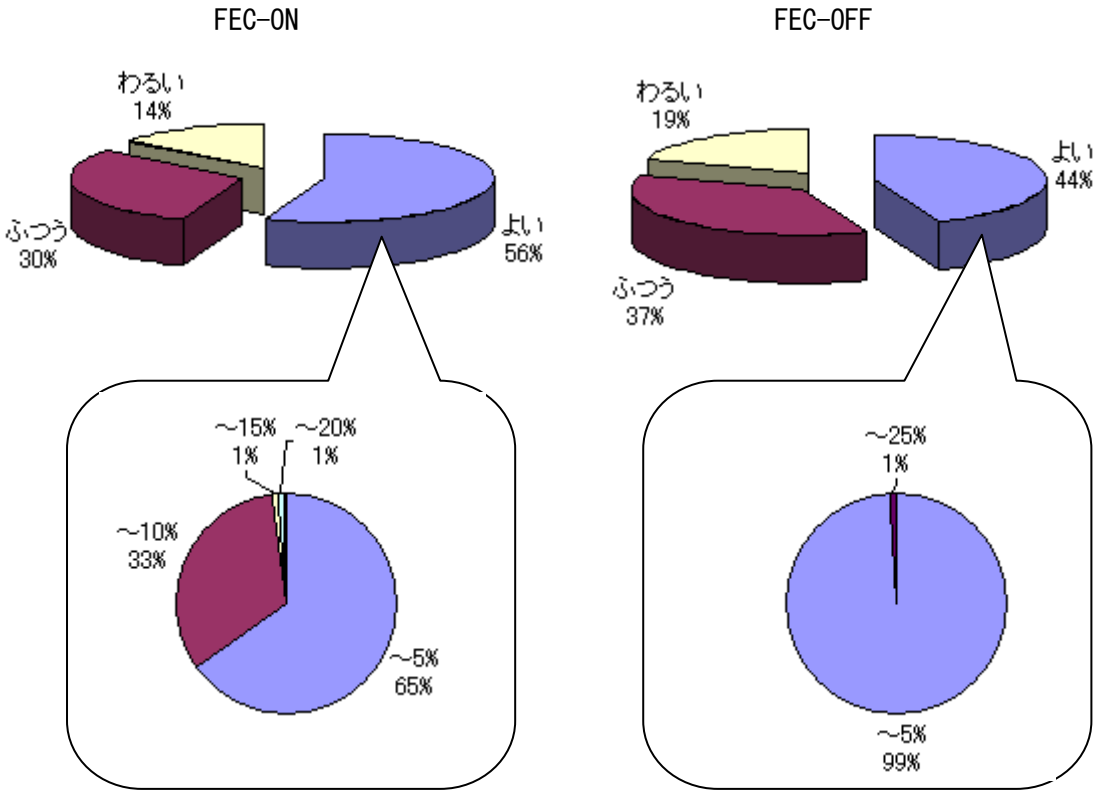


図 5-3-3-12 安定度が「よい」と答えたモニターのパケットロス率の内訳

(5) パケットロス率と視聴回数、安定度評価の関係を比較

安定度が「よい」と答えたモニターのパケットロス率と(3 分以上の)視聴回数、安定度評価の関係をグラフにしてみると、FEC-ON の場合(図 5-3-3-13)には、パケットロス 10%までに収まっており、そのうち、安定度が「よい」と答えたモニターの割合が半数以上であった。一方、FEC-OFF の場合(図 5-3-3-14)は、パケットロス 5%までに収まっており、それ以上パケットロスが発生してしまうと視聴に耐えられないため、FEC なしでの事業は困難が予想される。

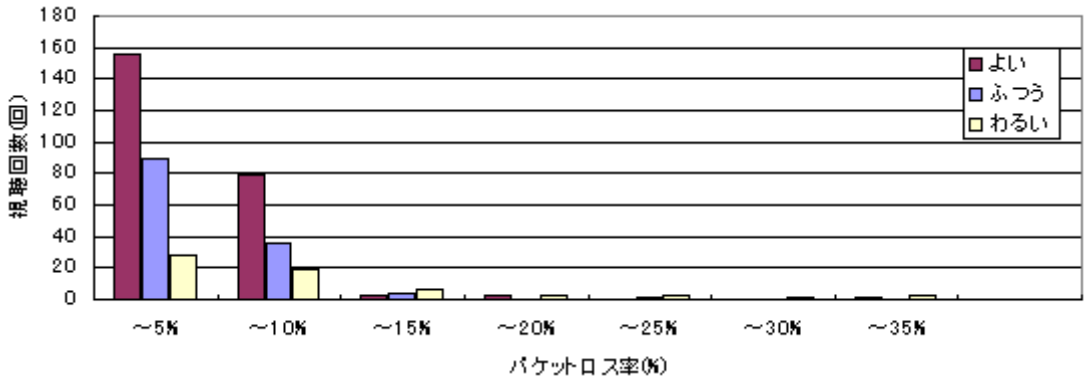


図 5-3-3-13 パケットロス率と視聴回数、安定度評価の関係(FEC-ON)

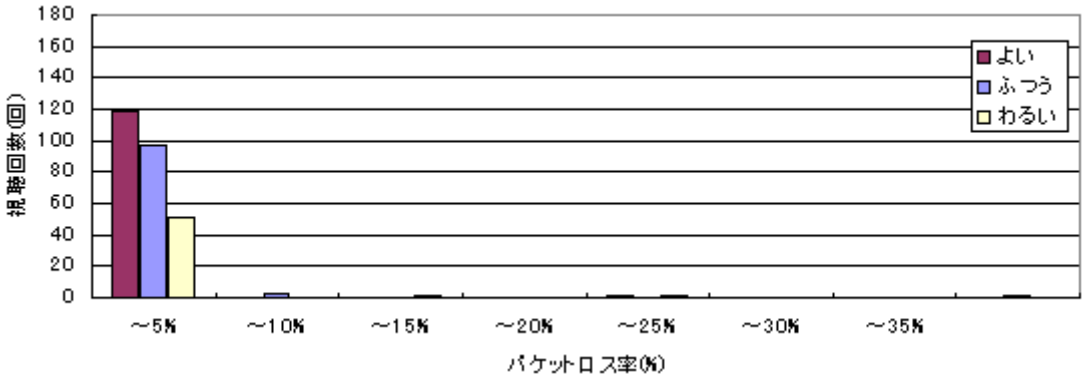


図 5-3-3-14 パケットロス率と視聴回数、安定度評価の関係(FEC-OFF)

(1)～(5)より、FEC 適用によって、「視聴回数の増大」、「視聴時間の拡大」、「満足度の向上」の効果があることが実証できた。

5-3-3-3　コンテンツに関するアンケート集計結果

アンケートでは、FEC 有無の比較以外にも、コンテンツに関する調査を行った。

・コンテンツに関するアンケート（平成 15 年 12 月 8 日～平成 16 年 1 月 8 日）

コンテンツの内容に関しては、半数以上のモニターが満足している。
コンテンツの料金に関しては、無料なら視聴するという人が多いのは容易に想像できるが、300 円でも視聴したいというユーザーが 13%いることに注目したい。

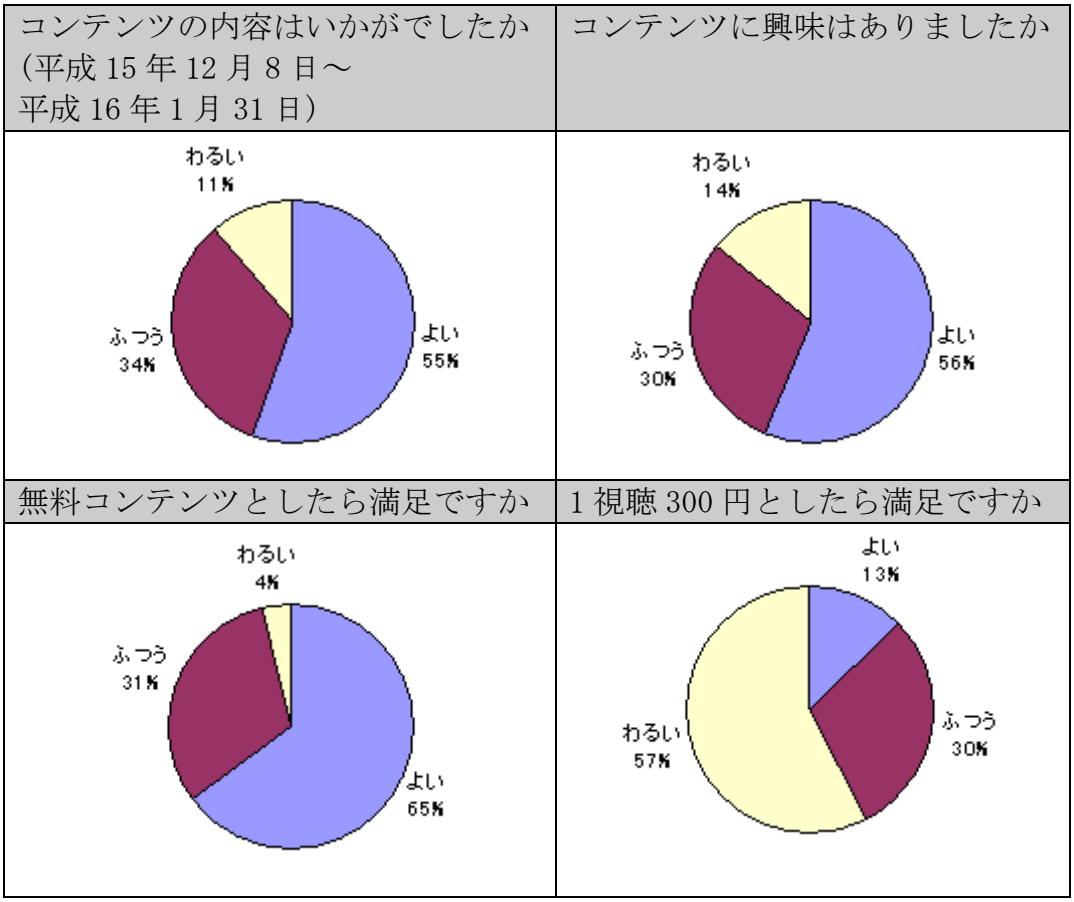


図 5-3-3-15　コンテンツに関するアンケート結果

5-3-3-4　視聴コンテンツ帯域(3M、6M)における利用回線種別の内訳

平成 15 年 12 月 8 日～平成 16 年 1 月 31 日のアクセスログから、3 分以上視聴したものを抽出し、3M・6M コンテンツの視聴者数の割合を見ると、3M は 54%、6M は 46%であり、3M の方がやや多い結果となった。このうち、3M・6M それぞれの利用回線種別の内訳を算出したところ、3M 視聴ユーザーの 74%が ADSL であり、6M 視聴ユーザーの 73%が B フレッツであった。また、6M 視聴の ADSL ユーザーの中でも ADSL8M が 1%であることから、帯域不足などで 6M が視聴できずに 3M を視聴している ADSL ユーザーが多いことが推測できる。ユーザーのネットワーク状況に合わせたコンテンツへの誘導が今後の課題である。

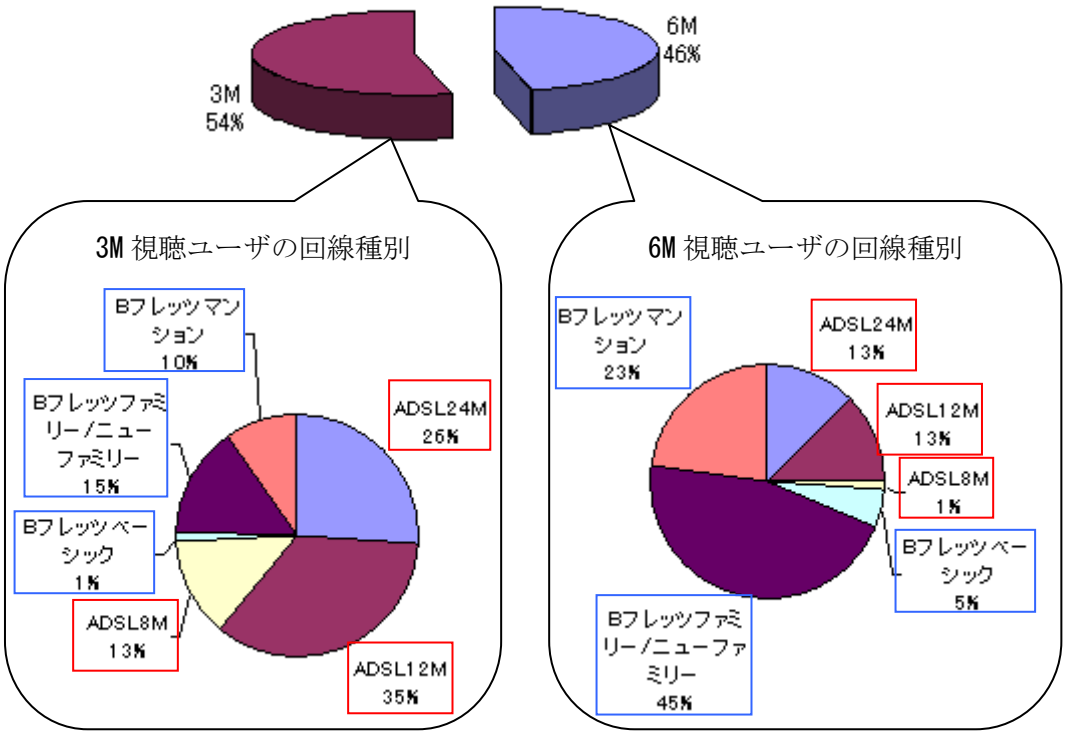


図 5-3-3-16　視聴コンテンツ帯域(3M、6M)における利用回線種別の内訳

5-3-3-5　利用回線種別のパケットロス率分布

　モニターのパケットロス率と視聴時間の関係を利用回線種別にグラフにしてみると、ADSL では、パケットロスが 35%までに散在しているが、B フレッツでは、8%までに収まっている。

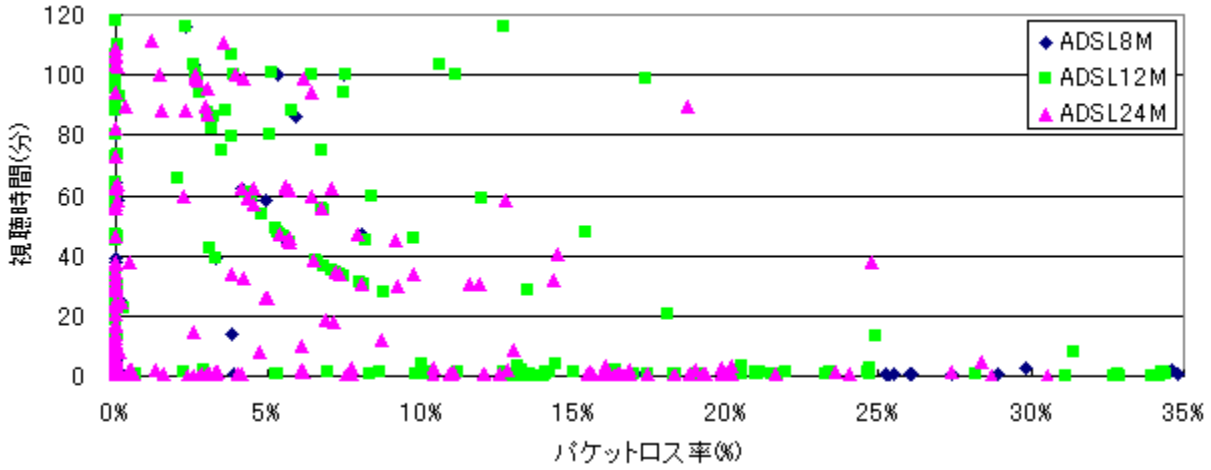


図 5-3-3-17　パケットロス率分布(ADSL)

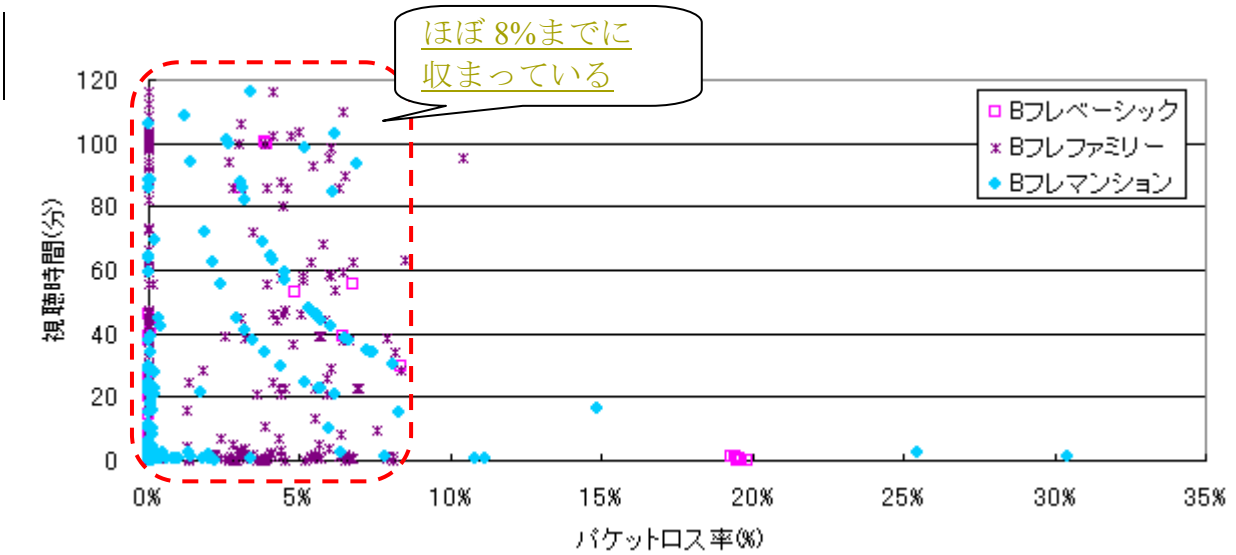


図 5-3-3-18　パケットロス率分布(B フレッツ)

5-3-3-6 Webアンケート

平成16年1月22日～平成16年1月29日に、Webアンケートを行った。アンケート結果を表5-3-3-1に示す。

回答総数226

表5-3-3-1 Webアンケート結果

STBの接続に（最低1回）成功しましたか	
1.はい	95.1%(215人)
2.いいえ	4.9%(11人)
成功していない理由は何ですか	
1.接続間違い	1人
2.モニター条件を満たしていなかった	1人
3.STB故障	1人
4.まだ試していない	0人
5.不明	5人
6.その他 ▪ IPアドレス取得に失敗 ▪ IPの取得に失敗しましたとの表示が出る ▪ モデムの問題?? ▪ 速度が遅い時がある ▪ 帯域が不足しているためトップページのみ	5人
視聴できたコンテンツは以下のどれですか？	
1.3M、6Mコンテンツ両方	68.2%(150人)
2.3Mコンテンツのみ	26.4%(58人)
3.両方とも視聴できなかった	5.5%(12人)
意図しない中断は発生しましたか	
1.なかった	59.6%(96人)
2.6Mは中断した	22.4%(36人)
3.3M、6Mともに中断した	14.3%(23人)
4.わからない	3.7%(6人)
視聴できない場合、以下の何れですか	
1.画面が真っ黒になる	54.3%(50人)
2.「サーバが応答しません」とエラー表示	13.0%(12人)
3.「混み合っている」旨のエラー表示	0%(0人)
4.わからない	14.1%(13人)
5.その他 ▪ 画面が乱れる ▪ 映像が途切れ途切れになる ▪ 帯域不足 など	18.5%(17人)
ネットワーク上でデータ抜けが発生しても安定した視聴ができる方法(FEC有)とそうでない方法(FEC無)の2種類で配信していますが、気づきましたか。	
1.気づいた	11.3%(24人)
2.気づかない	88.7%(188人)

気づいた理由は？	
1.3M、6M に関わらず安定しているコンテンツがある	60.7%(17 人)
2.3M はいつも安定しているが、6M は不安定	17.9% (5 人)
3.6M はいつも安定しているが、3M は不安定	10.7%(3 人)
4.その他 ▪ 3M でも視聴できるものとできないものがある。 ▪ 4.5 の速度では 6M は受信できない。 ▪ 時々ひどく画質・音質が悪いコンテンツがあったから。	10.7% (3 人)
気づかなかった理由は？	
1.どのコンテンツを選んでも安定している	48.2%(92 人)
2.どのコンテンツを選んでも不安定	12.0%(23 人)
3.視聴回数が少なくて気づかなかった	25.1%(48 人)
4.その他	14.7%(28 人)
3M と 6M で画質の違いを感じましたか	
1.6M の方がきれい	60.2%(97 人)
2.3M、6M で大差ない	28.6%(46 人)
3.その他 ▪ 6M は視聴できず、3M しか視聴していない(6 人) ▪ 6M しか視聴していない(7 人) ▪ MPEG-2 エンコードで 6M は少ない。最低 8M でエンコードしないと荒が出る	11.2%(18 人)
画質はよかったですか？ (TV と遜色ないか)	
1.よい (満足)	37.8%(79 人)
2.ふつう	43.5%(91 人)
3.悪い (不満)	18.7%(39 人)
画質が乱れた場合、どんな状況でしたか (複数選択可)	
1.映像にブロック状のものが混ざる	38.2%(128 人)
2.映像がコマ送りになる	18.5%(62 人)
3.映像が停止する	19.4%(65 人)
4.画面が真っ黒になる	15.5%(52 人)
5.その他 ▪ 乱れなかった ▪ 映像が粗くなる ▪ 特になし	8.4%(28 人)
画質が乱れた場合、どうやって復旧しましたか (複数選択可)	
1.自然復旧	52.5%(139 人)
2.リモコン操作 (早送り、巻き戻し、一時停止)	10.9%(29 人)
3.リモコン操作 (停止後、再度視聴)	15.1%(40 人)
4.電源オフ後、再度電源オン	12.1%(32 人)
5.その他 ▪ 乱れなかった ▪ 6M をやめて 3M に変えた ▪ PC でのインターネットを止めた	9.4%(25 人)

▪ 視聴を止めた	
音質はよかったですか？（TV と遜色ないか）	
1. よい（満足）	37.8%(79 人)
2. ふつう	56.9%(119 人)
3. 悪い（不満）	5.3%(11 人)
音質が乱れた場合、どんな状況でしたか(複数選択可)	
1. 音が飛ぶ	50.0%(117 人)
2. 雑音が混じって聞きづらい	14.1%(33 人)
3. 聞き取れない	7.7%(18 人)
4. その他 ▪ 乱れなかった ▪ 音声と画像が、ずれているコンテンツがあった ▪ 「キッ！」という鋭い金属音が一瞬出る ▪ 基本的に音量が低い為、テレビの音量を UP した	28.2%(66 人)
音質が乱れた場合、どうやって復旧しましたか(複数選択可)	
1. 自然復旧	53.2%(125 人)
2. リモコン操作（早送り、巻き戻し、一時停止）	6.0%(14 人)
3. リモコン操作（停止後、再度視聴）	9.8%(23 人)
4. 電源オフ後、再度電源オン	8.1%(19 人)
5. その他 ▪ 乱れなかった ▪ PC でのインターネットを止めた ▪ 視聴をあきらめた	23.0%(54 人)
音の乱れと映像の乱れ、どちらが我慢できますか	
1. 音	19.6%(41 人)
2. 映像	18.2%(38 人)
3. 両方我慢できない	57.4%(120 人)
4. その他 ▪ 無料なら映像の乱れは我慢できるが、有料だとお金を払うのですから両方我慢できない。 ▪ 音楽ものは映像、タレントイメージものは音。 ▪ 洋画なら音（字幕があるため） ▪ 両方ともあまり乱れなかったので我慢できる	4.8%(10 人)
視聴した際に、他の作業に影響を与えましたか(複数回答可)	
1. Internet 閲覧が遅くなった	41.0%(94 人)
2. メールの送受信が遅くなった	7.0%(16 人)
3. IP 電話が使えなくなった	3.1%(7 人)
4. その他 ▪ 特になし ▪ フレッツのセッションが足りなくなった ▪ インターネット、メールともに使えなくなった	48.9%(112 人)

5-3-3-7　Web アンケートの分析

(1) コンテンツの品質について

・視聴成功の割合

視聴できたか否かについてはほとんどのモニターが視聴に成功している。しかし、コンテンツ別で見ると 6M を視聴できたのは B フレッツ回線のモニターが中心で、ADSL 回線では 3M 止まりであり、速度の違いが顕著にあらわれた。画質については ADSL 回線利用のモニターのほうが「6M のほうがきれい」と答えているのに対して、B フレッツ回線のモニターは「大差ない」という回答が多かった。

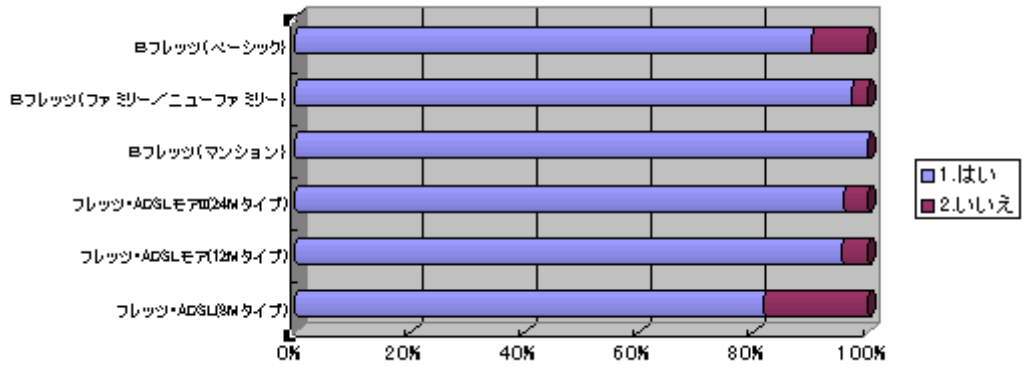


図 5-3-3-19　視聴成功の割合(回線別)

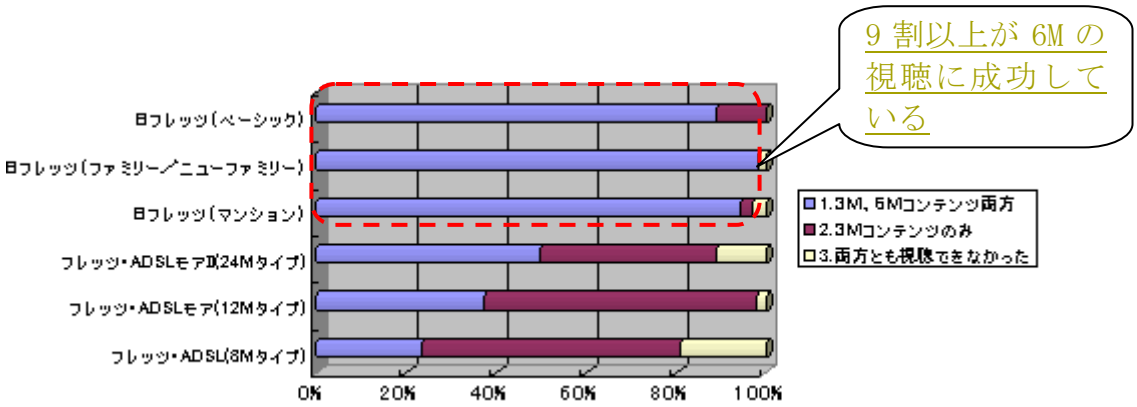


図 5-3-3-20　視聴できたコンテンツ(回線別)

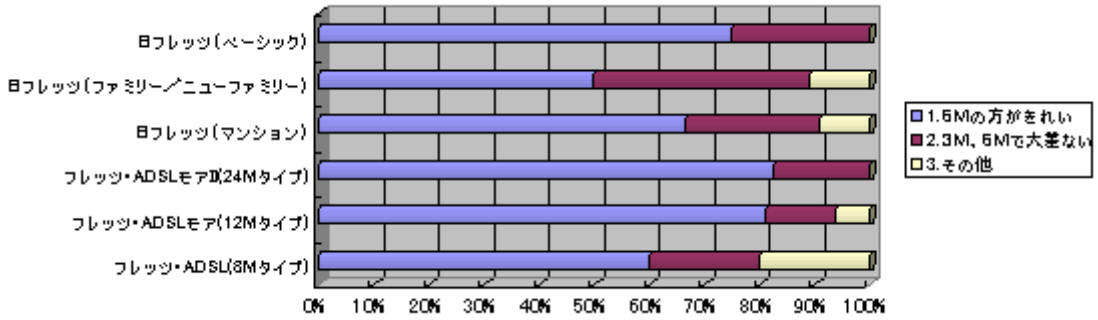


図 5-3-3-21　3M、6M の比較(回線別)

⑵ 画質・音質について

画質については、80%以上のモニターが、「よい」もしくは「ふつう」と答えている。一方、音質については、90%以上のモニターが、「よい」もしくは「ふつう」と答えている。画質よりも音質の方がやや満足度が高いようだ。

また、不満と答えたモニターは 10～20%程度と大変少なかったため、画質、音質については、問題はなかったと考えられる。回線環境によるバラつきも見られなかった。

音と映像の乱れのどちらが我慢できるかという問いに対しては、音のみ・映像のみと答えたモニターは、どちらも 20%弱と同程度で、両方我慢できないという回答が約 60%だった。これは、コンテンツにもよるところがあり、音楽コンテンツであれば、音が重要だが、洋画であれば字幕があるので、音は多少途切れても我慢できるとのこと。また、無料なら多少我慢するが、有料だったら、乱れは絶対に我慢できないという回答が見受けられた。有料サービスにした際には、品質保証が必須となるだろう。

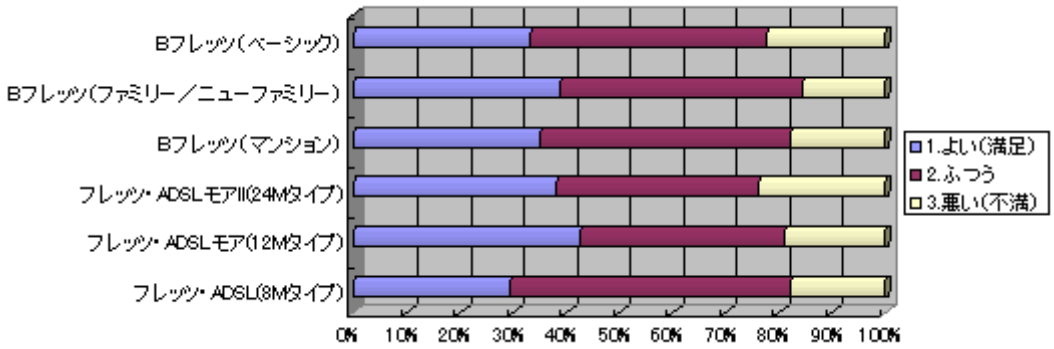


図 5-3-3-22　画質の満足度(回線別)

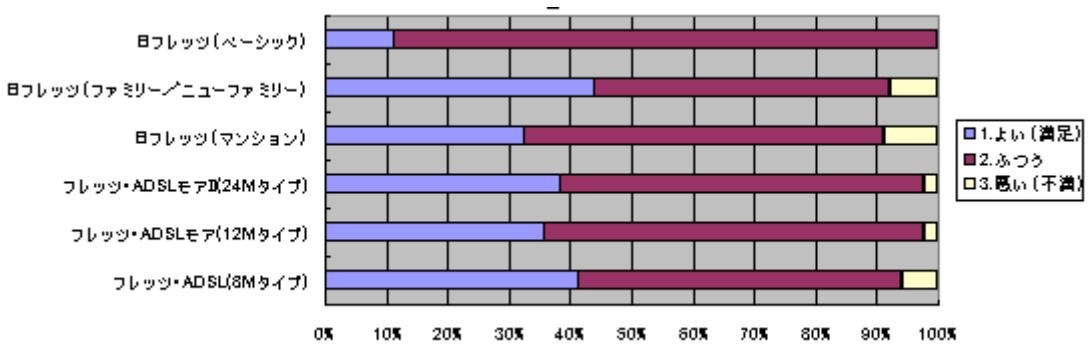


図 5-3-3-23　音質の満足度(回線別)

(3) エラー、不具合など視聴した結果発生した事象について

コンテンツ視聴中の中断発生は B フレッツ回線では少なく、ADSL 回線では多いとの結果となった。その他の不具合の事象としては「インターネットの閲覧が遅くなった」と「その他」の回答が多かった。その他の回答の大半は「特になし」「分からない」である。

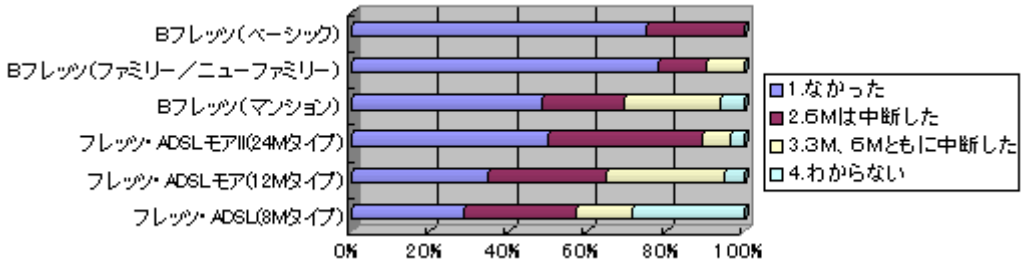


図 5-3-3-24　意図しない中断発生の割合(回線別)

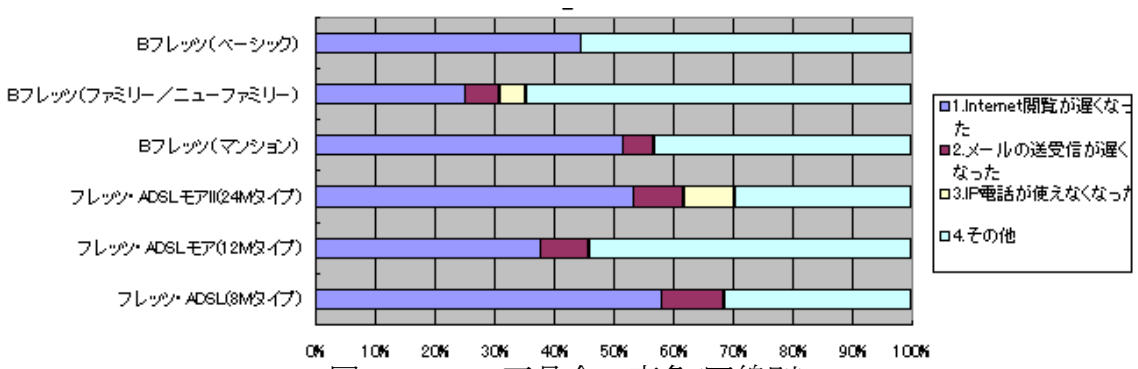


図 5-3-3-25　不具合の事象(回線別)

5-3-4　IETF への標準化活動

(1) FEC 機能付き RTSP プロトコル

映像伝送のための RTSP プロトコルに FEC 機能を付加したものを実装し、実証実験で検証を行い、その結果をもとにして IETF にて標準化の提案を行うという手順で標準化プロセスをすすめる。

提案するプロトコルは、RTSP を拡張し、ストリームサーバとクライアントの間で FEC パラメータの交換を可能にしている（以下、提案するプロトコルを「拡張 RTSP」と呼ぶ)。今回の実証実験では、FEC サーバを使用したが、提案するプロトコルでは FEC サーバなしの形態も考慮し、その場合 FEC サーバの機能はストリームサーバ自身に取り込まれることになる。

拡張 RTSP では、次のような方法で FEC パラメータの交換を実現している。

(1) クライアント（STB）は SETUP メッセージ中に FEC パラメータの記述を追加する。
パラメータ記述には、
Fec-method: method=<fec method>
Fec-param: <fec parameters>
という拡張ヘッダを使用し、ここに FEC パラメータの詳細を記述する。

(2) FEC サーバ（FEC サーバを使用しない形態では、ストリームサーバ）は、SETUP メッセージ中の FEC パラメータを解釈し、その応答をクライアントに返す。

(3) クライアントは、応答メッセージ中の FEC パラメータに対する応答を検出し、FEC を使用したストリーム伝送を行う。

実際のシーケンスを、図 5-3-4-1 および図 5-3-4-2 に示す。

なお、FEC サーバを使用した場合、拡張 RTSP はクライアントと FEC サーバ間で適用され、ストリームサーバから見れば通常の RTSP と変わらず上位互換性を確保するようになっている。

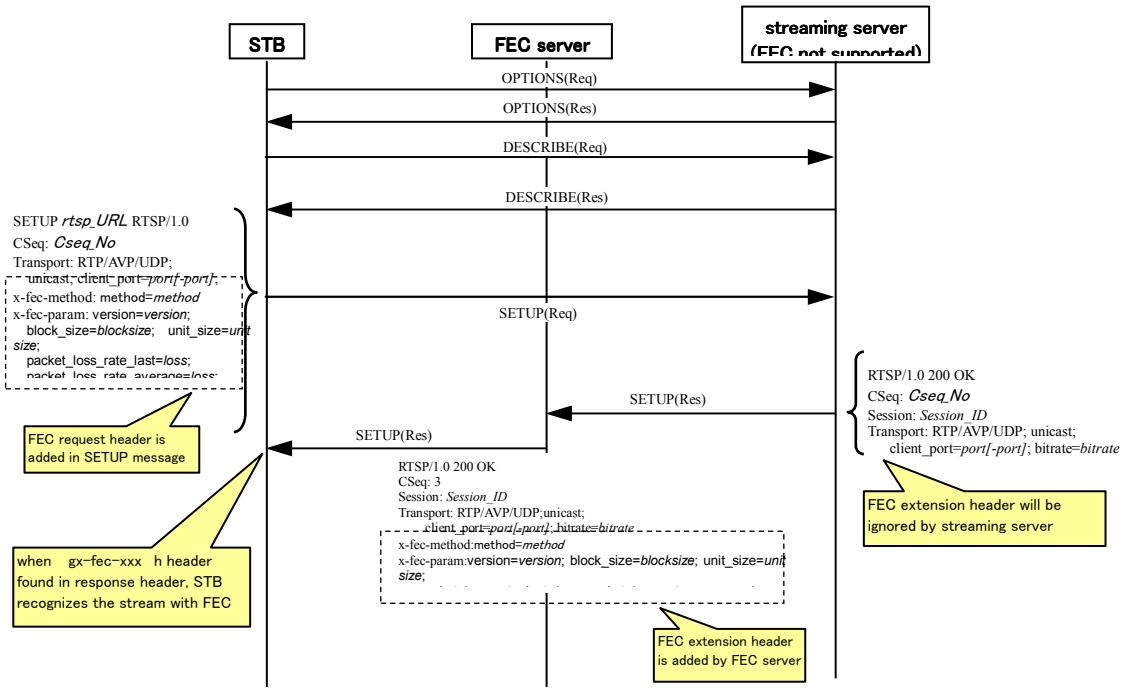


図5-3-4-1　拡張RTSPプロトコル（FECサーバあり）

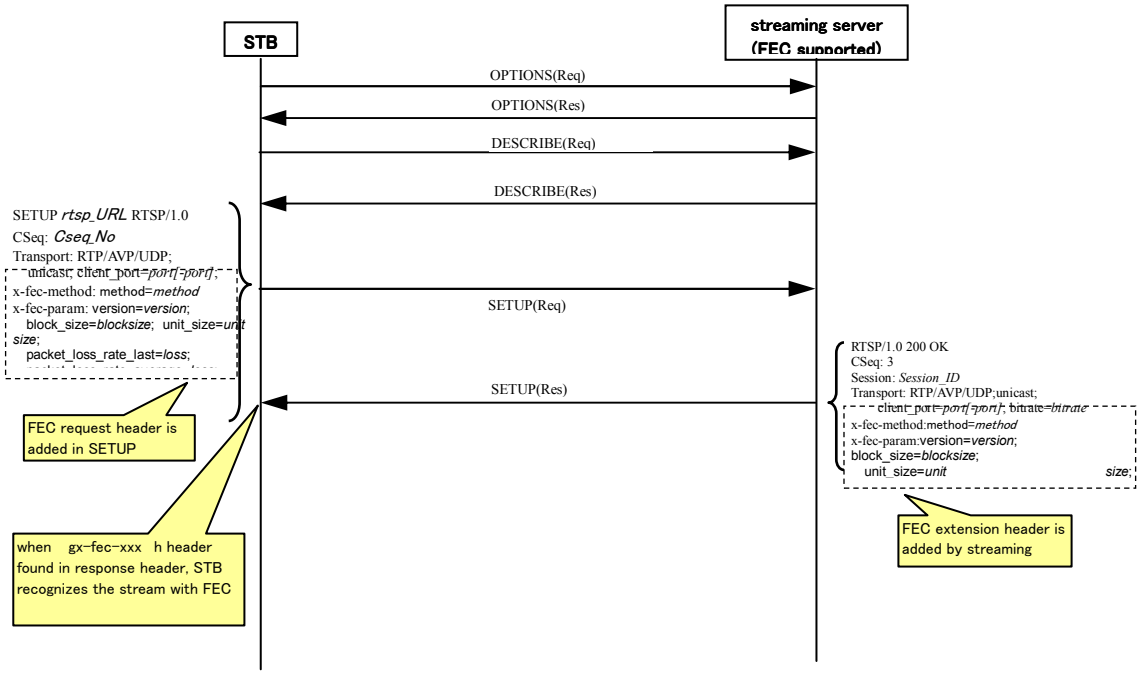


図5-3-4-2　拡張RTSPプロトコル（FECサーバなし）

(2) 拡張 RTSP の IETF ドラフト案

上記拡張 RTSP を元に関係者でレビューし、IETF に提案するための作業を進めている段階である。

5-3-5 結論

(1) FEC の有用性について

本実験では、FEC の有用性について、次のことが実証できた。

① FEC の効果によって、モニターの視聴時間が長くなる

FEC-ON の場合には、パケットロスが 10%以下であれば、多くのモニターが 20 分以上視聴していた。一方、FEC-OFF の場合は、パケットロスが 1%以上発生すると、視聴に耐えられず、ほとんどのモニターが視聴を止めてしまうことがわかった。

② FEC の効果によって、モニターの視聴回数が増える

FEC-ON の場合には、パケットロスが 20%以下でも 100 以上の視聴回数があるのに対して、FEC-OFF の場合は、パケットロスが１％以下でもほとんどアクセスがなくなってしまうことがわかった。

③ FEC なしでは、事業が成り立たない

FEC-ON では、約 10%までパケットロスが発生しても安定したサービスが提供できるが、FEC-OFF では、パケットロスが１％以下でも、サービスが実質不可能となってしまうことがわかった。

(2) ネットワーク構成と FEC 適用領域について

各種利用回線のモニターを抽出し、ネットワーク構成別に主観評価を行ったところ、6M を視聴できたのは B フレッツ回線のモニターが中心で、ADSL 回線では 3M 止まりであり、回線速度の違いが顕著にあらわれた。また、FEC が有効な場合、パケットロスが 5%発生した状況では、65%のモニターが安定度がよいと答え、パケットロスが 10%発生した状況では、33%のモニターが安定度がよいと答えた。つまり、パケットロスが 10%程度発生した状況でも FEC が視聴品質確保の一助となっていることがわかった。

(3) FEC 有無での操作性について

FEC-ON の操作性・応答性に関しては、30%近くのモニターが「よい」と答え、「ふつう」も含めると約 80%を占めている。また、FEC を適用することによって、トリックプレイ(早送りや巻き戻し)の操作性が向上した。

(4) アクセス傾向について

アクセス数は、総アクセス数、同時アクセス数ともに週末に多い傾向にある。同時アクセス数のピーク数は、14 クライアント(全体の約 5%)であった。実際のサービスを提供するにあたって、これらを考慮したネットワーク・サーバ構成が必要になる。

(5) コンテンツについて

コンテンツの内容に関しては、半数以上のモニターが満足している。画質・音質については、不満と答えたモニターは 10～20%程度と大変少なかったため、問題はなかったと考えられる。回線環境によるバラつきも見られなかった。画質・音質の乱れについては、無料なら多少我慢するが、有料だったら絶対

に我慢できないという回答が見受けられた。有料サービスにした場合には、品質保証が必須である。

(6) STB サービス全体について

サービス全体の感想としては「いくつか気になる点はあるが、概ね良い」が圧倒的に多く、この視聴形態自体は受け入れられることが分かった。

- 今後の課題

本実験では、FEC 適用率は全てのモニターに対して一律であったが、FEC 適用率を ADSL と B フレッツで変えたり、ユーザーのネットワーク状況に合わせるといったことが考えられる。

5-4　帯域抑制手段に関する研究、中継装置用超高速低遅延順方向誤り訂正デバイスの試作・実験

5-4-1　序論

デファクト標準化を念頭においた、順方向誤り訂正機能およびその選択制御機能のデバイス化と評価プラットフォームとしてのサーバ・クライアントおよびホームゲートウエイ等低速低価格中継装置の試作を行なう。

5-4-2　研究内容

インターネット上でのコンテンツ配信に用いられているプロトコルは、TCP/IP が一般的で、伝送途中で発生したパケット欠損を、ARQ (Automatic Repeat reQuest)という再送方式で補っている。UDP を用いたリアルタイムの映像伝送を行う場合は、伝送路途中で発生するパケット欠損に対しては、前方誤り訂正技術 (Forward Error Correction : FEC) を用いて、受信側(前方)で欠損パケットを補う方法が用いられる。この理由は、再送による遅延時間の予測が難しいためである。FEC を用いると、想定された数以下のパケット欠損であれば、受信側で元のパケットを復元することが出来る。しかし、元のパケットを復元するためには、一定時間に余分なパケットを送信する必要があり、伝送速度の増加によってルータの処理帯域を圧迫してしまい(輻輳が発生し)、パケット欠損が増加する。ルータでパケット欠損が発生すると、TCP 系のパケットは、受信確認が出来ず、結果的に伝送速度が低下する(輻輳制御)。一方 UDP 系のパケットには、帯域を抑制す機能がついていないため、結果的に TCP 系の通信を押しのけて通すことなる。インターネット上の均衡を保つためには、UDP パケットに対しても何らかの抑制メカニズムを考えておく必要がある。この際、課金モデルやインターネット上の仕掛けに対する考察も、必要となる。

5-4-3　IETF の動向

IETF（Internet Engineering Task Force)では、継続時間の長い UDP 系の通信に輻輳制御機能を付加しようという動きがある。

（１）DCCP (Datagram Congestion Control Protocol)

DCCP はまだ RFC にはなっておらず、作業中であるが、IETF のホームページにいくつかのドキュメントが提出されている。基本的には、輻輳制御機能をも

ったトランスポート層プロトコルとして DCCP を作り、UDP に替わる方式を普及させていこうとしている。再送制御はしないものの、受信側から送信側に受信確認を返すことで、送信側で輻輳状況を判断し、送信を抑制するメカニズムである。具体的に、２つのタイプが提案されている。

(A) TCP-like な方式

TCP と同様に Windowing (受信確認を受け取る前に送信可能なデータ量を制限すること) を行い、このサイズをパケット欠損によって制限していくやり方。短時間での送信レートがパケット欠損によって急激に変動する性質があり、ストリーミング映像配信には向かない。

(B) TCP-Friendly な方法

パケット欠損率を測定し、その欠損率における TCP の性能を推定して、それと同程度に平均送信レートを調整する方法。送信レートの変動が平滑化されるので、ストリーミング映像配信に適している。この方式は (TCP Friendly Rate Control : TFRC)は、RFC3448 に詳しい記述があり、アルゴリズムが示されている。この中に、パケット欠損率から TCP 性能（スループット）を算定する式が示されており (出典："Modeling TCP Throughput: A Simple Model and its Empirical Validation", Proc. ACM SIGCOMM 1998")、パケットの往復にかかる時間 (往復遅延) (Round Trip Time : RTT) とパケット欠損率によって TCP 系に適用されるの伝送速度が算出される。パケット欠損率と TCP Friendly な伝送速度の関係を、図 5-4-1 に示す。この結果から、100ms 程度の往復遅延がある場合には、パケット欠損率 0.1%以上では、映像伝送速度 4Mbps で伝送することが出来ない。

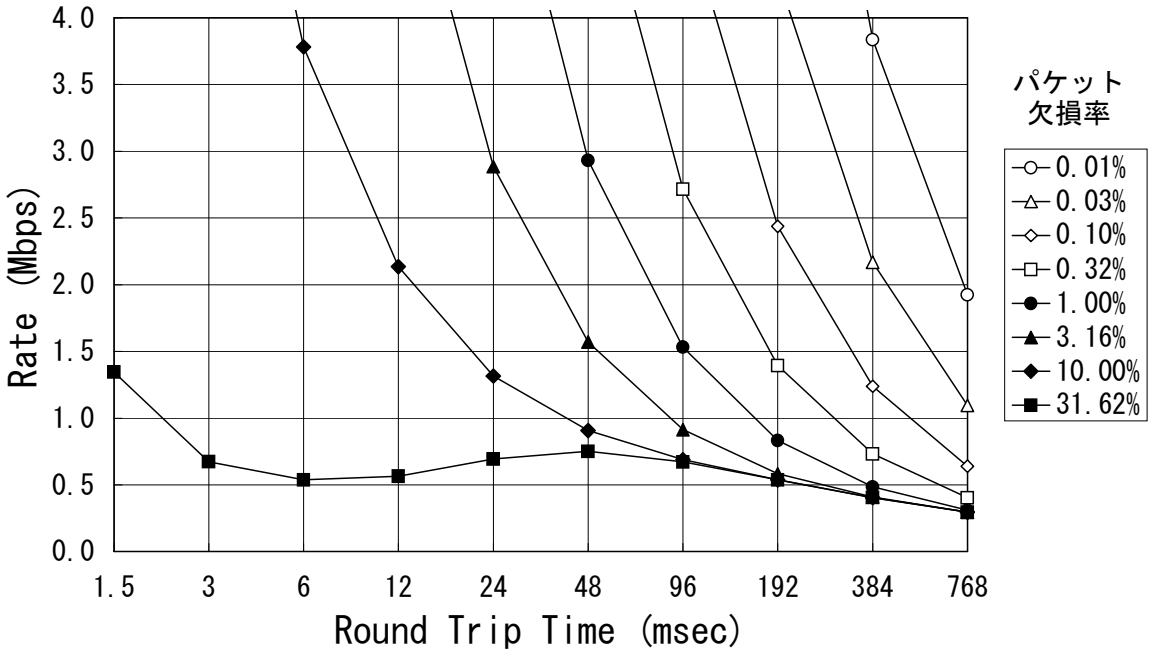


図 5-4-1　パケット欠損率と TCP Friendly な伝送速度の関係

またパケット欠損率：1%，RTT：100ms の状況では、伝送速度の 1.5Mbps と算出される。この状態で、映像伝送速度 4Mbps を確保しようとすれば、基準となる伝送速度より 2.5 倍程度の伝送帯域を過剰に確保していることになる。

TFRC に基づく計算では、インターネットでの映像配信の速度を増加させようとすると、インターネット上でのパケット欠損率を低減する必要がある。つまり「映像配信の速度：増加」→「パケット欠損率：低減」→「FEC による欠損パケットの復元」→「映像配信速度：増加」という正帰還がかかり伝送速度が発散してしまうため、FEC を用いる事が出来なくなる。しかし、ベストエフォート型の定額サービスでは、何らかの抑制メカニズムが必要であり、その基準の考え方として、この方法は意味があると考えられる。

ベストエフォート型の定額サービスで上記の方法を用いる場合には、ユーザの契約内容に応じて、基準となる伝送速度の何倍までの帯域を許すかという設定を事前に行う方法が考えられる。この許可された帯域内に映像伝送速度が収まるように、サーバ(FEC エンコーダ)側で映像伝送速度を動的に調整する。映像伝送速度が、許容された帯域より大きい場合には、映像自体の伝送速度を低下させる方法も、考慮する必要がある。

5-4-4　結論

TCP 系の配信と UDP 系の配信を、インターネット上で混在させる場合の方法について調査を行い、TFRC という方法が有効であるとの結果を得た。

5-5　総括

誤り訂正符号に関する理論的な検討から、Digital Fountain 社の誤り訂正符号が有力であるとの結論を得た。実機で、Reed-Solomon 符号と Raptor 符号の処理速度の比較を行った結果、Raptor 符号が有利であるとの結論を得た。インターネット上のパケット欠損とジッタの関係をモデル化し、その関係について検討を行った。また、商用ネットワーク上で FEC を用いた映像配信の実証実験を行い、FEC の有効性を確認した。

参考資料、参考文献

[1] Michael G. Luby, ″Information Additive Code Generator and Decoder for Communication Systems,″ US Patent No. 2001/0019310.

[2] M. Amin Schkrollahi, Soren Lassen, and Michael Luby, ″Multi-Stage Code Generator and Decoder for Communication Systems,″ US Patent No. 2003/0058958.

（添付資料）

１　研究発表、講演、文献等一覧

・学会発表

(1) ″インターネット映像配信における Reed-Solomon 誤り訂正符号と Multi-Stage 符号との比較検討″, 森田 哲郎, 西本 裕明, 佐々木 隆志, 高橋 豊, 電子情報通信学会 信学技報 CQ2003-36.

(2) ″XOR-based coding に基づく効率的なマルチキャスト映像配信制御の提案″, 高橋潤, 戸出秀樹, 村上孝三, 電子情報通信学会 信学技報 NS2003-209 PN2003-37.

(3) ″サポート配信機能を有するメタコンテンツのマルチキャスト配信制御″, 久野和英, 高橋潤, 戸出秀樹, 正城敏博, 村上孝三, 電子情報通信学会 信学技報 NS2003-210 PN2003-38

(4) ″Erasure Code を用いたマルチサーバ配信方式の検討″, 藤枝 太一, 久野 和英, 高橋 潤, 戸出 英樹, 正城 敏博, 村上 孝三, 電子情報通信学会 2004 年 総合大会

・論文発表（査読あり）

(1) ″Comparison of Loss Resilient Ability between Multi-Stage and Reed-Solomon Coding,″ Tetsuo Morita, Hiroaki Nishimoto, Takashi Sasaki, and Yutaka Takahashi, 11th International Conference on Telecommunication Systems.

(2) ″Performance Analysis of HC/RC for Rich Content Distribution,″ Kaesinee Thongsisod, Takashi Sasaki, Tetsuo Morita, and Yutaka Takahashi, 11th International Conference on Telecommunication Systems.

２　再委託研究報告書

最新の高速低遅延順方向誤り訂正技術の国際的調査研究とネットワーク上の多段利用を想定したクライテリアの確立
(京都大学大学院 情報学研究科 システム科学専攻 高橋研究室)

ネットワークのモデル化とシミュレーション技術の開発ならびに高信頼性コンテンツ配信プロトコルの研究

（大阪大学大学院 情報科学研究科 情報ネットワーク学専攻 村上研究室）

以上