平成 12 年度

学士学位論文

非対称通信ネットワークを用いた マルチキャストコンテンツ配信方式

Multicast Scheme using Asymmetric Satellite Communication Networks

1010366　　伊藤 雄

指導教員　　清水 明宏

2001 年 2 月 5 日

高知工科大学 情報システム工学科

要 旨

非対称通信ネットワークを用いた
マルチキャストコンテンツ配信方式

伊藤 雄

広域性という特徴を持つ衛星回線を用いたデータ配信方式において，学校などへの教材配信サービスについて考える．学校環境では，授業単位でのコンテンツ利用が主であるため，利用時間や利用コンテンツがある程度似通っていると考えられる．そのため，個々の受信者が別々にファイルの要求を行う現在の使い方ではなく，効率よく受信者にファイルを配信できる，衛星通信回線を利用したマルチキャスト配信方式に特に注目した．

そこで，本検討では，高速で高信頼なマルチキャスト配信方式として期待されている，RMTP [*1] という通信プロトコルを用い，実証実験を通して，RMTP の有効性を検証する．まず，衛星通信回線と地上回線を使用した非対称通信ネットワーク上で実際に配信を行い，送信レート，配信データのファイルサイズ等を変化させた際，それぞれのスループットへの影響を測定した．その結果，送信レート毎のスループットへの影響や，理論的に得られている，配信データサイズやファイル配信方法毎のスループット特性が，実験的にも得られるという結果を得た．

キーワード　衛星通信，非対称通信ネットワーク，マルチキャスト同報配信，RMTP

[*1] RMTP: Reliable Multicast Transport Protocol

通信の途中で受信者からの応答確認を集約し，その中の再送要求に応答するという手順で通信を行う，レイヤ 4 のプロトコル．第 3 章 ”高信頼マルチキャスト転送プロトコル”，6 ページを参照． [5]

Abstract

Multicast Scheme using Asymmetric Satellite Communication Networks

Takeshi Ito

Satellite communication networks are suitable for wide-area destribution and effective for sending teaching materials to schools. And, in especially the school emvironment, it's possibility of accessing same contents simultaneously. Therefore, multicast scheme is valid for this situation. In this scheme, the data are destributed to recivers effiently, but not required by each recivers separately.

Then, the effecticeness of RMTP is applied to asymmetric satellite communication networks contains satellite for insent and internet for decent was evaluated.

The caracteristics of thoughput was measured by destribution experiments using asymmetric satellite communication network. As the result of destribution experiments measured by changing some parameters, some fundamental data - influence packet-loss and destribution-time depending on changing transmission rate, filesize, and some parameters - for the best throughput was obtained.

key words satellite communication, asymmetric networks, multicast scheme, RMTP

目次

図目次

表目次

第 1 章

はじめに

分散処理型のコンピュータネットワークが発展するにつれ，複数のノードへ同じデータを同時に送信し，効率を向上させる要求が高まってきている．

一対一の通信より，一対多，多対多の通信になると，同じデータを同時に複数の端末へ送信する需要がさらに増加すると考えられる．さらにこの先，マルチメディアアプリケーションが発展すれば，こういった需要は強くなってくる．そこで，データを必要としている端末にのみ送信する，マルチキャスト同報配信方式が特に注目されている．

本検討で扱う衛星通信回線は広域性や同報性という特徴がある．衛星通信回線は，どのような地域にも高速なデータ配信を行うことができる特徴と，複数の拠点に同時にデータを配信を行うことができることから，学校への教材配信サービスに有効である．また，この特徴は，同時刻に同データの必要性が生じるような，特殊な環境において，非常に有効である．特に学校環境を考慮すると，学校では，授業単位でのコンテンツ利用が主であるため，利用時間や利用コンテンツがある程度似通っていると考えられる．

第 2 章

検討の背景

本検討では，実証実験を通してデータによる定量的な分析，評価を行い，マルチキャスト同報配信による高速高信頼マルチメディア情報配信技術の確立を目指す．また，UDP [*1]の上位に位置し，通信ミドルウェアとして高信頼ファイル配信を可能とする RMTP を使用し，高信頼マルチキャスト配信における最適スループット及び QoS[*2]に関する検討を行う．

2.1 衛星を用いた非対称通信ネットワーク

非対称ネットワークとは，上り方向と下り方向の回線が違う，非対称なネットワークのことである．本検討のシステムは，衛星を用いた非対称通信ネットワーク環境を用いる．これは，高速の衛星通信回線を下り方向に利用し，上り方向に地上回線を使用した非対称通信による高速大容量のファイル配信を実現する地域衛星イントラネットワーク環境のことである．

本検討で用いる衛星通信回線は広域性や同報性で特色のある回線であり，学校教育や生涯教育などに役立つと考えられている．しかし，現在では，衛星回線を利用して，個々が別々のファイル要求を行うような現在の使い方を取っており，本来の衛星通信回線としての特徴を生かしていない．そこで，衛星を使ったネットワークでは，多数の端末に同時にデータを配信するということを有効に生かす使い方について検討する．

例えば，利用時間や利用形態がある程度似通っている学校環境特有の状況下において，コ

[*1] UDP: User Datagram Protocol

信頼性のないデータ通信を提供する，レイヤ 4 のプロトコル．付録 A を参照．

[*2] QoS: Quality of Service

ンテンツの同時利用の可能性があることから，センタ側手動の同報配信が可能なマルチキャスト配信システムの有効性が期待されている．これは，コンテンツの同時利用により，同じデータが各要求に対して，個別に送られることになってしまうため，ネットワークに対する負荷がある程度同時刻に偏ってしまうことになる．これを解決するコンテンツ配信方式としてマルチキャスト配信方式に注目した．

2.2　同報配信

複数の受信側にデータを受け渡す伝送方法として，ユニキャスト，ブロードキャスト，マルチキャストがある．まずユニキャストでは，データをコピーしてそれぞれを各受信者に送信する方式である．次にブロードキャストでは，データパケットをネットワークの全範囲に送信する．これらのデータは受け取る必要のある端末が少ない場合でも全端末に送信される．そしてマルチキャストと呼ばれる方式では，最初に一つのパケットだけが送信され，途中必要な時にだけネットワークがパケットを複製する．

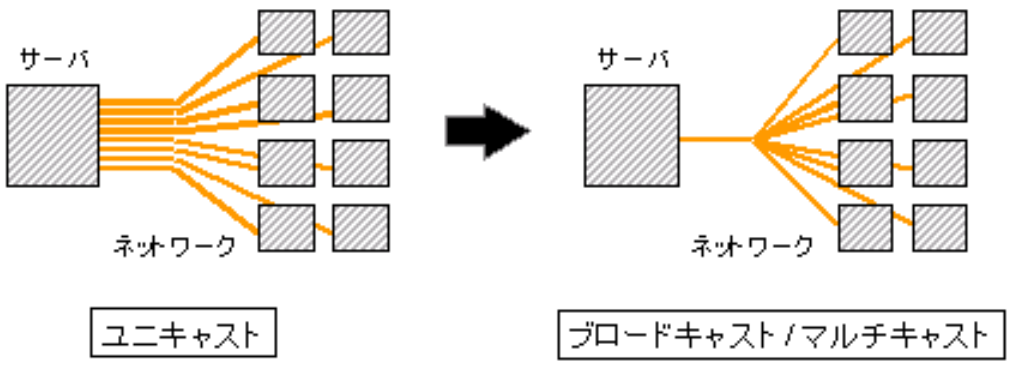

図 2.1　同報配信

図 2.1 に示すように，ユニキャスト方式による配信と，ブロードキャスト方式やマルチキャスト方式による配信を比べた際，ブロードキャスト方式やマルチキャスト方式が，よりネットワーク負荷を大幅に抑えることができることがわかる．

2.3　マルチキャストの有効性

ユニキャスト同報配信では，つまり，一対一の通信では，データを送る端末の数だけ送る必要があり，その分ネットワークに負荷がかかってしまう．それに対し，マルチキャストやブロードキャストでは複数の端末に対して同時にファイルを配信するのに向いている．ブロードキャスト同報配信では，ネットワーク上すべての端末にデータ配信を行う送られる．従って，配信するデータとはまったく関係のないホストであってもデータの受信を行わなければならず，その分よけいな負荷をかけることになってしまう．それに比べ，マルチキャストによる同報配信では，受信を意図する端末の集合にのみデータを同時に送信することができるため，送信の必要な拠点に効率よくデータを送る事ができ，ネットワーク負荷も抑えることが可能である．このため，マルチキャストを使った配信方式の有効性が検討されている．

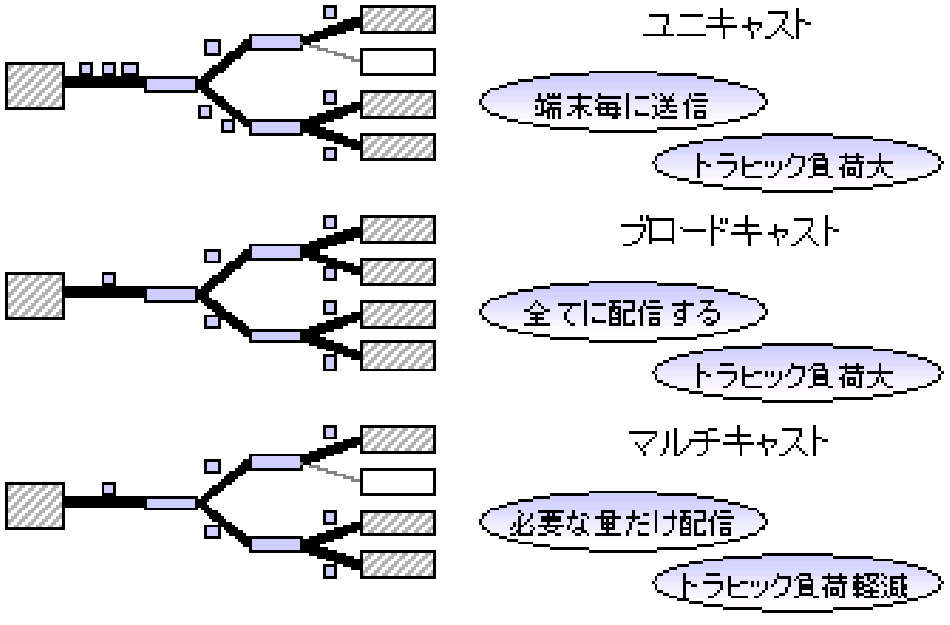

図 2.2　ユニキャスト/マルチキャスト/ブロードキャスト

現在，高信頼なマルチキャスト同報配信として提案されている方式では，全データを一括

で送信し，その後全クライアントから送達確認を受信し，再送要求されたパケットのみ同報で再送する．このようなマルチキャスト通信方式では，複数のクライアントから一度に多数の送達確認が発生するため，サーバでの確認処理に負荷を与えることになる．よって，処理負荷をかけず高いスループットが得られる高信頼マルチキャスト通信方式を検討する必要がある．そこで本検討では高信頼な同報配信を行うため，RMTP を用いた実証実験を行い，このプロトコルの有効性を検討する．

第 3 章

高信頼マルチキャスト転送プロトコル

RMTP[*1] [5] は，電子新聞や営業情報などのマルチメディア情報を，同時に多数のユーザに効率よく配布するための技術である．このプロトコルは，日本アイ・ビー・エム株式会社東京基礎検討所と日本電信電話株式会社 (NTT) 情報通信検討所との共同検討プロジェクトで開発された．確実にデータを送信する為に様々なパラメータが定められており，配信ファイルに最適なパラメータを実証実験において抽出することによって高信頼なデータ配送を行うことができる．

3.1 RMTP の有効性

RMTP は，通信の途中で受信者からの応答確認を集約し，その中の再送要求に応答するという手順で通信を行う レイヤ 4 のプロトコルである．受信の際，欠落パケットがあった場合は欠落パケットをマルチキャストのパケットとしてグループに再送される．そして各受信者は，ユニキャストで応答確認を送信して，送信者がこの要求に答える．送信者は，受信者側のパケットの欠落がなくなったという通知があるまで再送を続行する．この方式の場合，確認信号のフィードバックがあふれる可能性が残るとしても，プロトコルの開発者はこの応答確認と再送要求を制御することで，約 10,000 人の送信者からの応答処理が出来ることが確認されている．

[*1] RMTP: Reliable Multicast Transport Protocol

3.2　RMTP の通信処理

以下に，RMTP のファイル送信処理を示す[*2].

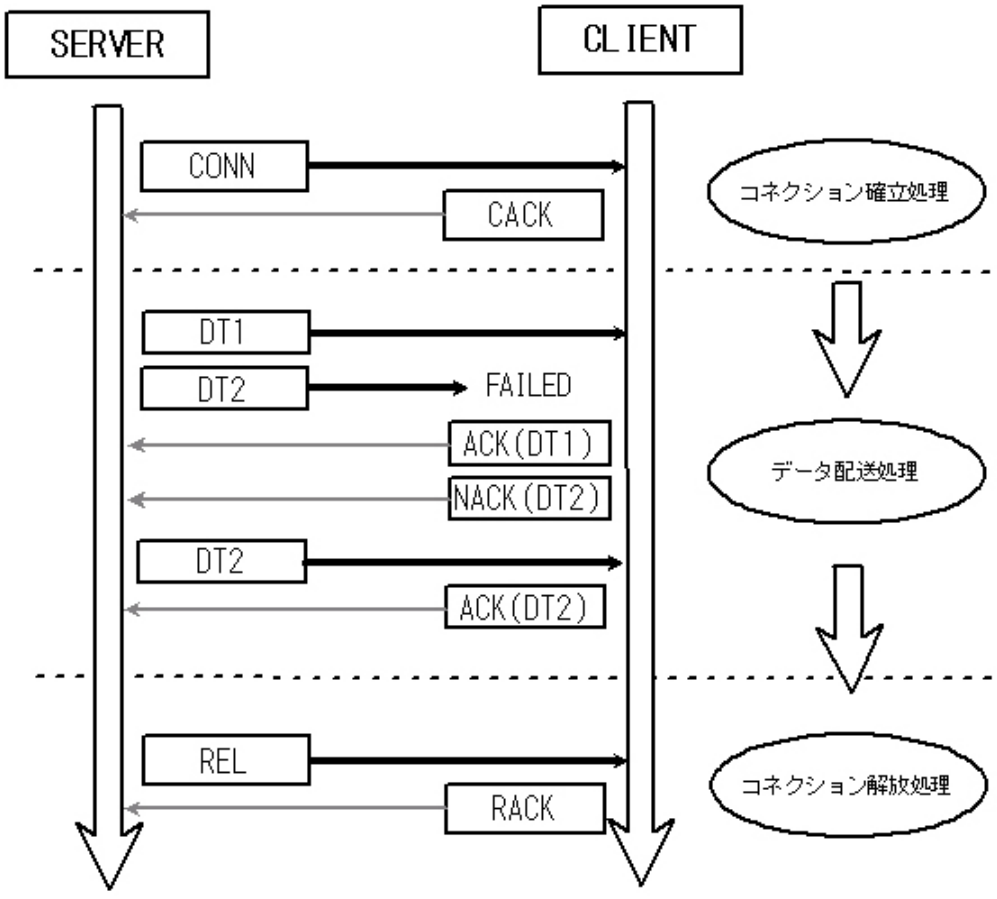

図 3.1　RMTP の通信処理

RMTP サーバのファイル送信プロセスでは，コネクション確立，マルチキャスト送信，コネクション開放の 3 段階がある.

1. 前処理 （コネクション確立）
2. 配送処理（マルチキャスト送信）
3. 後処理 （ コネクション開放）

これら「前処理」,「配送処理」,「後処理」の 3 段階にかかる総時間が，ファイル送信にか

[*2] 各 RMTP パケットの概要は，付録 C に添付.

かる総通信時間となる．

■前処理（コネクション確立）

まず，コネクション確立では，サーバはクライアントに対してファイル送信の開始 (CONN) を通知する．ここで，送信するファイルのサイズ，ブロックの大きさなども通知する．次に，通知を受けたクライアントはコネクション確立応答（CACK) をサーバに応答する．サーバにはあらかじめ受信クライアントのリストが保持されており，コネクション確立時にはこのリストに基づいて適正なクライアントからの応答かどうかも判断される．

■配信処理（マルチキャスト送信）

次に，マルチキャスト送信のプロセスでは，ブロックに分割されたファイル（DT）をマルチキャストで同報送信を行う．ファイル全体を送り終わった時点で，クライアントからの受信確認を待ち，クライアントは，コネクション確立時に得たファイルについての情報から，全ての受信が終わったかどうかを確認する．不足している部分（ブロック）があれば，その不足ブロックのシーケンス番号を通知し再送をサーバに要求する．サーバは全てのクライアントからの受信応答（ACK) や再送要求通知（NACK) を取りまとめ，再送要求のあったブロックを，再度マルチキャストで同報送信する．

再送を要求していたクライアントはこのマルチキャストを受け取り，全て受信完了となれば，受信応答（ACK) をサーバに通知する．データに不足している部分がある場合，再度，再送要求通知（NACK) をサーバに行う．再送手続きは，原則として全てのクライアントから受信通知（ACK）があるまで繰り返される．ただし，サーバが持っている受信クライアントリストにないクライアントからの再送要求は無視される．

■後処理（コネクション開放）

最後に，コネクション開放のプロセスでは，サーバはクライアントに対し，ファイル送信の終了（REL) を通知する．通知を受けたクライアントは終了通知（RACK) をサーバに応答し，受信手続きを完了する．

第 4 章

検討内容

非対称通信ネットワークに RMTP を利用した，マルチキャスト同報配信について，実証実験を通してデータによる定量的な分析，評価を行い，高速で高信頼なファイル配信技術の確立を目指す.

4.1 検討目的

衛星を使った非対称ネットワークや，同報配信方式については，従来から検討が続けられてきており，シミュレーションによって最適な方法が検討されている [1].

しかし，その評価はシミュレーション上のものであり，実際にデータを得ているというものは少ない．本検討では高信頼なマルチキャストデータ転送を実現する RMTP を導入する．また，RMTP を用いたマルチキャスト配信実験はまだ行われていない．そこでこのプロトコルの有効性実証実験によって検証する.

4.2 検討項目

本検討ではデータ配信実験を行うことにより，基礎データの取得を目的とする．検討項目を 2 つに分類した.

- スループットの調査
- 高信頼性の評価，検討

それぞれの検討項目を以下に示す.

■スループットの調査

- ファイルサイズによるスループットの変化の測定から配信に関する問題点抽出

  通常ファイル / 大容量ファイル / 超大容量ファイルの 3 種類の通信を行い，スループットや再送回数を取得する.
- 配信方法の効率性に対する調査

  複数の小さなファイルを配信する際に，そのままの形で配信する場合と，いったんアーカイブして配信する場合とで，どちらのスループットがよいかを調べる.
- 高速配信に影響するパラメータの抽出

  抽出したパラメータより，スループットを最速とするための配信用パラメータセットを実証実験により求める.

■高信頼性の評価，検討

- 高信頼の評価，検討

  ファイルサイズやファイルの種類（画像，静止画，音楽など）によって送達率（再送回数，応答確認など）の変動について，机上および実証実験によって伝送エラーに対する送達率評価し，高信頼性を検討する.
- 障害発生時の対策

  スループットの低下，配信失敗端末の発生など，受信に障害が発生した際，どのような対策を取っておけば影響を少なく出来るかを検討する.
- 他の通信環境との比較

  他通信プロトコル (TCP) を用いた時，他環境 (CATV 網) の時のスループットとの比較，検討
- 高信頼マルチキャスト転送プロトコルの QoS に関する分析，評価

実験項目として以下 3 点を挙げる．

1. 高信頼マルチキャスト転送プロトコルの QoS に関連する各種要因の検討
2. コンテンツ蓄積装置～配信サーバ，配信サーバ～受信クライアント

   コンテンツ蓄積装置に登録されたコンテンツファイルが配信サーバへ転送されるまで，および，配信サーバから受信クライアントへの配信処理に要する時間を別々に取得する．
3. コンテンツ蓄積装置～受信クライアント

   コンテンツ蓄積装置へのコンテンツファイル投入から，実際に受信クライアントへファイルが配信されるまでの時間を測定し，最適化を行う．

4.3　ネットワークシステム構成

本検討の実験環境では，品川（東京）に RMTP サーバ装置を設置し，サーバからは衛星回線を用いて各受信端末へデータを配信する．今回，衛星からのマルチキャスト配信データ受信は以下の高知県下 4 地点で行われる．また，衛星回線を利用した下りのマルチキャスト配信データは各受信拠点にある RMTP クライアント装置に直接送られる．

- 南国オフィスパーク
- 青少年センタ
- ふくし交流プラザ
- 吾川村教育委員会

第 5 章

実証実験結果，考察

本検討では，RMTP を用いて実証実験を行った．まず，配信する際，スループットに影響するパラメータを変化させることで，RMTP の信頼性を確認した．

5.1　通信ログ解析

本検討の実験では，データを配信したときの通信ログ[*1]にて解析を行う．そこで，通信ログを解析しやすくするため，テキスト形式の通信ログを CSV 形式に変換するプログラム[*2]を作成した．RMTP の通信ログでは，取り出すべき項目の順番や場所などが決まっている．このプログラムでは，キーワードによって単語を検索し，必要な数値を取り出て書き換えることで形式を変換している．

5.2　机上検討と実験結果

5.2.1　通信処理の各段階における応答性能

ファイル配信プロセス[*3]を以下に示す．

図 5.1 に示すように，前処理，配送処理，後処理にかかる時間の総時間が，ファイル配信にかかる総時間となる．まず，各データ送信プロセスに対して，以下のことが考えられる．

[*1] 付録 D に添付.

[*2] 付録 E に添付.

[*3] 3.2 節，7 ページを参照.

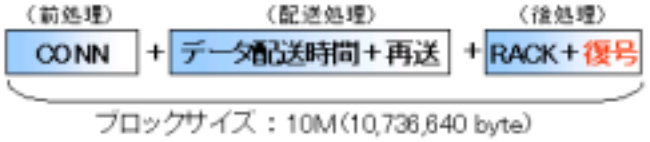

図 5.1　配信プロセス

- 前処理の段階では，コネクション確立のパケットを送り，受信側が応答信号を返すまでの処理を行う．これは正常に送ることができた場合，転送ファイルサイズに関係なく一定値を取ると考えられる[*4]．
- 配送処理の段階では，受信側で最初のデータパケットを受信してから最後のパケットを受信し，受信完了応答を返すまでの処理を行う．これはデータの転送にかかる時間であり，データサイズに依存する．
- 後処理の段階では，受信側がデータをすべて受け取った後，サーバからのコネクション解放通知に対して，受信側が開放応答を返すまでの処理を行う．これは正常にコネクション開放の処理が行われた場合，転送ファイルサイズに関係なく一定値を取ると考えられる．

以上から，スループットに影響を与えるのは配送処理にかかる時間と考えられる．しかし，各段階での通信途中のパケット紛失や配信失敗などがあることから，また，受信端末数が増加することにより応答パケットの集約が増加することから，様々な要因がスループットへ影響を及ぼすことが予想できる．これは，端末数が増えることでデータが正常に配信されない端末が増えることにより，再送に伴って配送処理にかかる時間が増加するためである．

また，コネクション確立やコネクション開放ができない端末がある場合，設定上の待ち時間が経過するまで，サーバが待機する．応答待ち時間をどれくらいとるかということが，最適なスループットを検討する上で重要である．

[*4] 配信実験より，約 1 秒かかるということを確認した．

5.2.2　送信レートとスループット

配信される 1 ブロック毎のスループットを調べるため，10M バイトのファイルを配信し，送信レートを変化させた際の，配信時間，配信処理に対する実レート，パケット紛失率を測定した．また，受信端末数による影響を考慮し，単一端末への配信とした．

また，本検討で用いる衛星回線の帯域は，最大 約 250K バイト/s であり，RMTP 配信プログラムの性質上，200K バイト/s 以下の時，転送レートはほぼ一定の，約 130K バイト/s を取る．よって，この実験では 200K バイト/s から，10K バイト/s ずつ，260K バイト/s まで増加させて，ファイルを配信した．

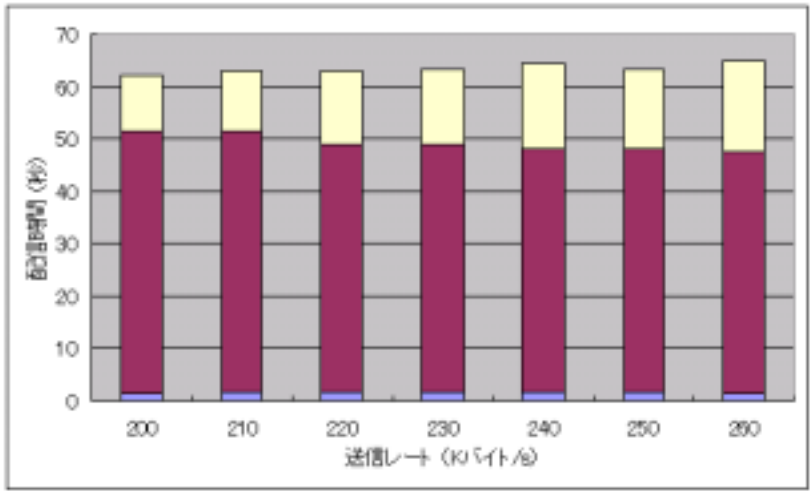

図 5.2　送信レートと配信時間

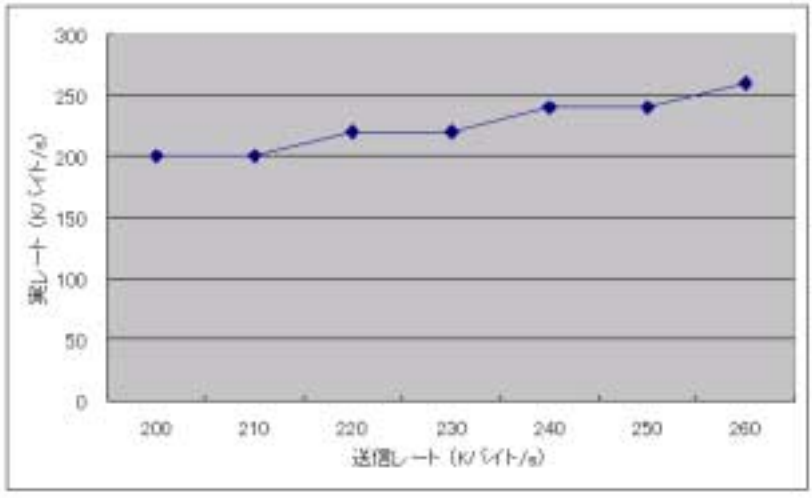

図 5.3　配信処理に対する実レート

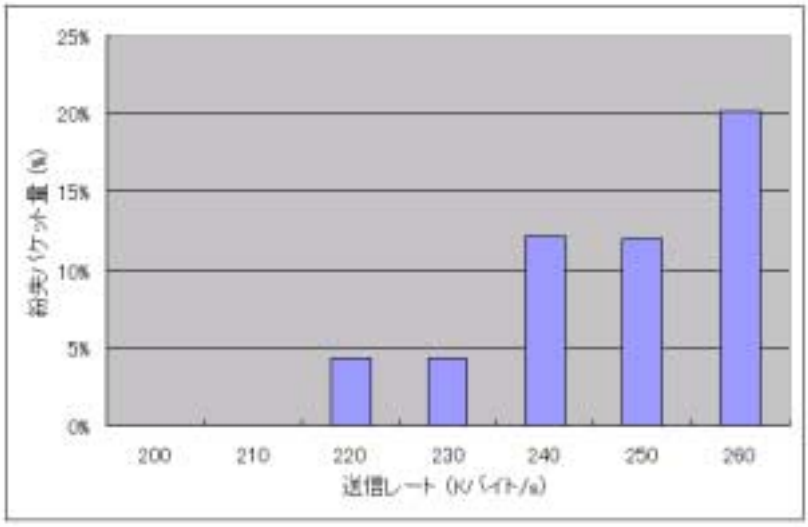

図 5.4　送信レートと紛失率

図 5.2，図 5.3，図 5.4 に示すことから，以下のことが言える．

- 送信レートによる全体の配信時間への影響はほとんどない．
- 実レートが 20K バイト/s 毎に増加している．
- 220K バイト/s 以上のレートではパケット紛失が発生する．

これは，送信レートを上げることによって発生した，パケット紛失に伴う再送処理により，全体の配信時間を増加させたためと考えられる．その結果，配信処理に対する実レートを下げる要因となった．

5.2.3　ファイルサイズとスループット

ファイルサイズを増加させた際の，スループットへの影響を測定するため，600M バイトまでの大容量ファイルを配信し，配信時間，パケット紛失率を測定した．また，ファイルサイズによる影響を測定するため，図 5.4 の結果から，送信レートによるパケット紛失への影響がなかった 200K バイト/s において配信を行った．また，受信端末数による影響を考慮し，単一端末への配信とした．

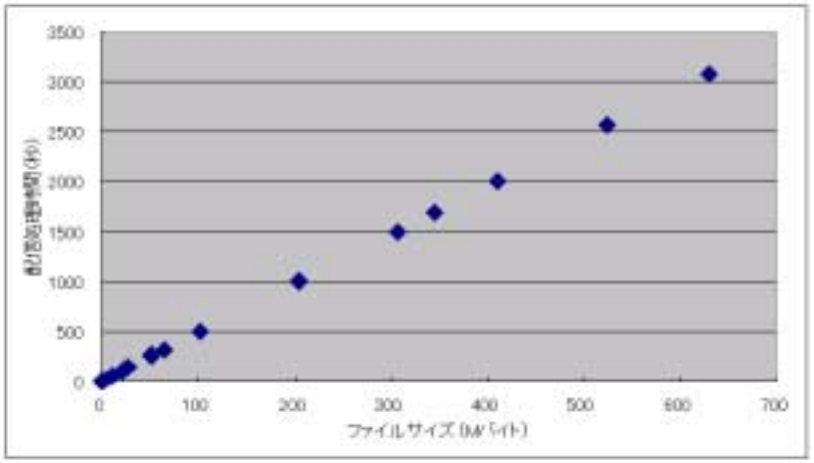

図 5.5　ファイルサイズと配信時間

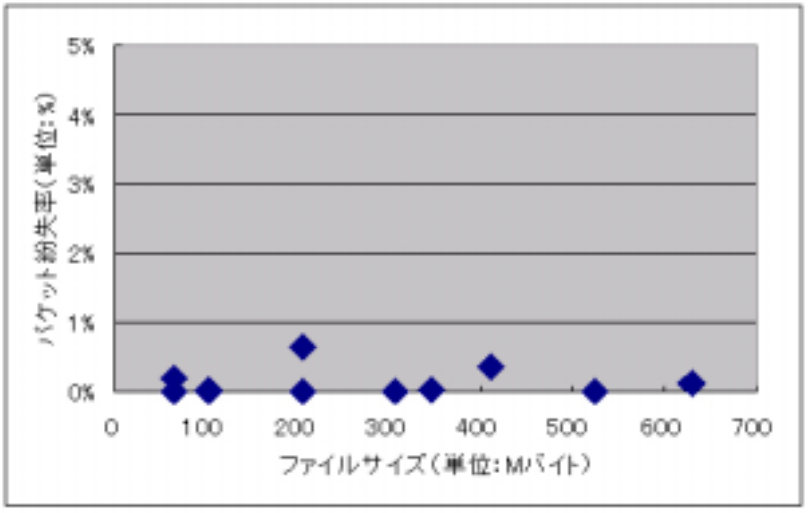

図 5.6　ファイルサイズとパケット紛失率

図 5.5 に示すように，配信にかかる時間がファイルサイズに比例していることから，RMTP では，ファイルサイズに関わらず，安定した配信が行えるということがわかった．

このとき，この傾きは 約 200K バイト/s となった．

また，図 5.6 に示すように，送信レート 200K バイト/s においては，ファイルサイズに関わらず，パケット紛失率が 0.7% 以下となるという結果を得た．このことから，200K バイト/s において，RMTP では，パケット紛失を抑えた，効率のよい配信が行えるということが言える．

5.2.4　ファイル配信方法とスループット

図 5.1 に示すように，配信ファイルを細かく分割して配信する際，各プロセスに対して以下のことが考えられる．

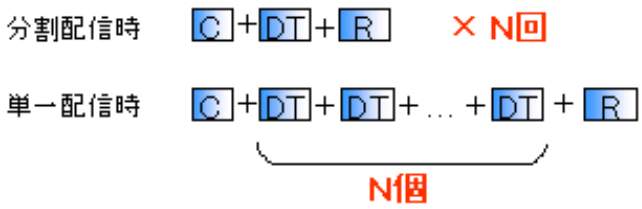

図 5.7　分割方式と配信プロセス

- 前処理

  配信ファイルを単一で配信した場合，前処理におけるコネクション確立は最初の 1 回のみとなる．それに対して複数のファイルとして送ったとき，コネクション確立を分割ファイル毎に行う必要があるので，分割する数だけ前処理にかかる時間が増加する．
- 配送処理

  配信ファイルの合計サイズは分割方式に関係なくほぼ同一であるから，配送処理では分割方法による影響はほとんどない．
- 後処理

  前処理の場合と同じく，分割する数だけコネクションの開放を行う必要があるので，その分後処理にかかる時間が増加する．

図 5.7 に示すように，複数の細かいファイルを配信するより，単一のファイルとして配信すれば，よいスループットが得られると期待される．

そこで，分割するサイズを変化させた際の，スループットへの影響を測定するため，100M バイトのファイルを以下の分割サイズにおいて配信し，それぞれの配信時間を測定した．また，分割するサイズによる影響を測定するため，図 5.4 の結果から，送信レートによるパ

ケット紛失への影響がなかった 200K バイト/s において配信を行った．また，受信端末数による影響を考慮し，単一端末への配信とした．

以下に，この実験で測定する際，使用した分割サイズ，ファイル分割数を示す．

分割サイズ	分割した数
10M バイト	10 個
20M バイト	5 個
50M バイト	2 個
100M バイト	1 個

表 5.1　分割サイズと分割数

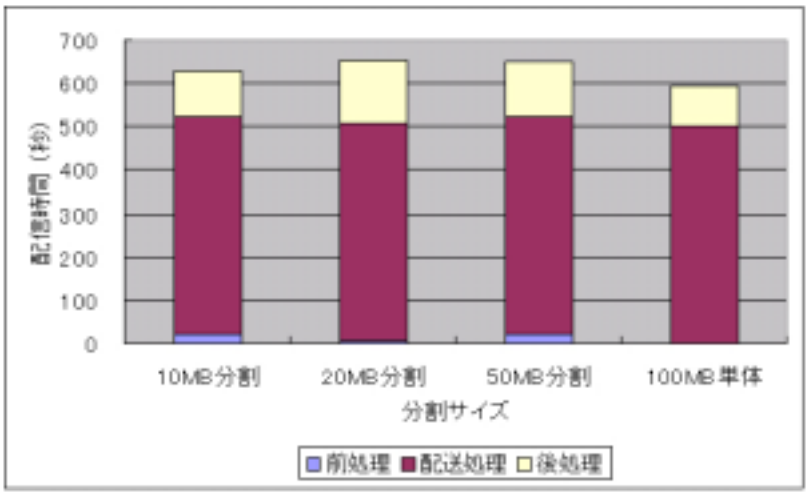

図 5.8　分割方法の違いと配信時間（100M バイト配信時）

図 5.8 に示すように，今回の実験データでは，50M バイト分割において開放処理に時間がかかったことから，予想した実験データ結果を得ることができなかった．

100M バイトを配信した結果，分割サイズによる配信時間およびスループットの影響は，確認されなかった．これは，コネクション確立の回数の増加が配信時間に影響を与える，さらに細かい分割を行った際，影響すると考えられる．

今後は，さらに，他の分割サイズを用いた実験により，分割サイズのスループットへの影響を明らかにしていく必要がある．

5.2.5　復号処理

受信完了を，配信されたファイルが受信者側で使用するところまでと考えると，受信者側の記憶装置[*5]への書き込みが終わった時点で，ファイル配信の終了といえる．この場合，配信自体の終了とは違い，オペレーティングシステムによる記憶装置への書き込み処理が終わった時点，すなわち，ファイルの最終更新が終わった時刻を計ることで測定できる。

ファイルを配信した際，ファイルサイズの復号処理時間への影響を測定した結果，本検討で使用した端末[*6]において，以下のような結果が得られた．また，この数値は，受信端末の性能に依存する．

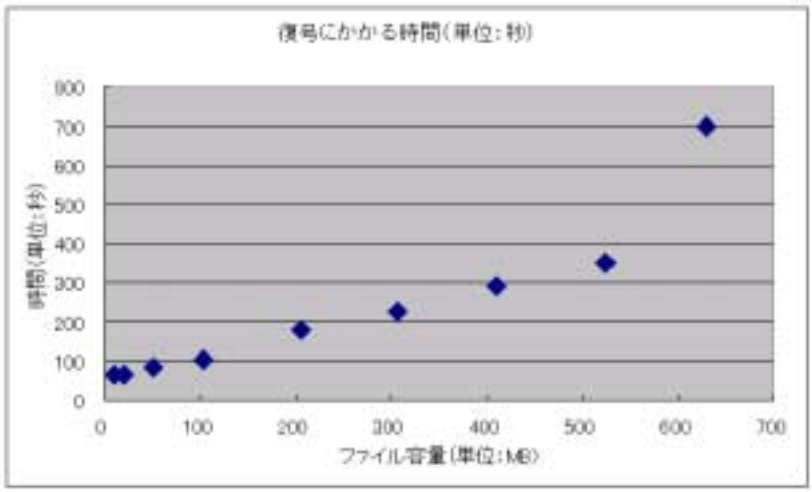

図 5.9　復号にかかる時間

図 5.9 に示すように，ほぼ配信ファイル容量に比例して書き込み時間がかかっている．600M バイト配信時に復号時間が多くかかっているのは，使用端末の構成上，書き込み処理のための容量が不足していたためと考えられる．

[*5] この場合，ハードディスク

[*6] 付録 B に添付.

5.2.6　受信端末数とスループット

受信端末数が増えることで，各配信プロセス 3.2 において，スループットへの影響が予想される．

- 前処理において，コネクション確立が失敗した端末の発生が増加する．また，再度コネクション確立を行い，それに伴ってすべての端末へのコネクション確立完了までの時間が増加する．
- 配送処理において，データの受信が失敗した端末の発生が増加する．また，受信失敗に伴う個別再送のため，配信時間に影響が出る．
- 後処理において，コネクションの開放が失敗した端末の発生が増加する．また，正常に開放されない場合，開放通知が届くまでサーバが待機するため，処理時間が余分にかかる．

また，送信の必要な拠点に効率よくデータを送る事ができる，マルチキャスト配信方式という性質[*7]から，受信端末数のスループットへの影響は，ほとんどないということが予想される．

この実験は，今後の課題として，受信端末数を変化させた実験により，パケット紛失率，配信時間への影響等，受信端末数のスループットへの影響を明らかにしていく必要がある．

[*7] 2.3 節，4 ページを参照.

第 6 章

まとめ

まず，RMTP の評価を行うため，データを非対称通信ネットワーク上で配信する実証実験を行った．その結果，以下の事項が明らかになった．

1. 送信レートが 220K バイト/s 以上になると，送信レートに比例してパケット紛失が増加するため，全体の配信時間が短縮されない．
2. 配信にかかる時間がファイルサイズに比例して増加していることから，RMTP では，ファイルサイズに関わらず，安定した配信が行える．
3. 送信レート 200K バイト/s において，パケット紛失率は 0.7%以下となる．
4. 分割/単一 の方式で配信を比較したとき，100M バイト配信時では影響は出ない．

今後は，さらに，他の配信パラメータを用いた実験により，RMTP の有効性を明らかにしていく必要がある．

謝辞

本検討を行うに際し，貴重な御助言や多大なる御指導を頂いた本学情報システム工学科の清水 明宏 助教授に深く感謝いたします．

また，本研究室院生 井上 富幸氏，岡田 実氏，田鍋 潤一郎氏ならびに本研究室学部生 大石 恭裕君，岡崎 友輝年君，小橋 誠治君，篠原 直之君，竹内 紀貴君，谷藤 喜彦君，林 竜也君，間城 昌厚君，安岡 隆司君，山崎 愛さんに感謝いたします．さらに在学中，親切なる御助言を頂いた諸先生方に心より感謝いたします．

参考文献

[1] 藤部，他：マルチメディア衛星通信における高信頼マルチキャスト通信方式の提案と評価，電子情報通信学会，SAT98-26，1997

[2] 北原, 他：TCP/IP の衛星通信回線への適用について, 電子情報通信学会, SST2000-11, SAT2000-21(2000-06)，2000

[3] 秋山，他：衛星利用データ配信システムの評価，情報処理学会，マルチメディア通信と分散処理 80-29，1997

[4] 小川，他：衛星通信回線における TCP の性能評価に関する実験的検討，電子情報通信学会，SAT99-77，CQ99-21(1999-07)，1999

[5] RMTP ホームページ，http://www.trl.ibm.co.jp/projects/rmtp/

[6] Dave Kosiur マスタリング TCP/IP IP マルチキャスト編，オーム社，1999

[7] Thomas A. Maufer，楠本訳：bit 別冊 IP マルチキャスト入門，共立出版，2000

付録 A

OSI 7 階層モデル

OSI 7 層構造モデルとは，ISO[*1]が提唱している，Open System Interconnection[*2]を表す．この階層構造を以下に示す．

階層番号	名称	規定している内容
7	アプリケーション層	通信者の目的とするサービス
6	プレゼンテーション層	通信する情報のコード化，暗号化
5	セッション層	送信する権利の制御など
4	トランスポート層	送受信での到達確認など
3	ネットワーク層	通信路の選択，設定，開放
2	データリンク層	一区間での誤り検出など
1	物理層	通信速度，電圧など

表 A.1　OSI 階層のモデルの各機能

[*1] ISO: International Standard Organization - 国際標準化機構

[*2] 開放型システム間相互接続

付録 B

使用端末構成

本検討で用いた衛星系データ受信装置の構成を示す.

項目	構成
CPU	Pentium　500MHz
OS	日本語 WindowsNT Workstation4.0
メモリ	128MB
ハードディスク	10GB（空き容量 1GB）
ネットワーク，インタフェース	Ethernet 10BASE － T ／ 100BASE － TX 1 ポート
外部接続インタフェース	SCSI
チューナ機能	CS 用衛星の周波数を選択可能
衛星信号変換機能	衛星信号から IP パケットへの変換が可能
CS アンテナ	CS 用受信アンテナ、径 60cm

表 B.1　衛星系データ受信装置構成表

付録 C

RMTP 通信パケット一覧

RMTP の通信処理で用いられるパケットの一覧を示す.

Packet	概要	Communication Sequence
DT	データパケット	a server to receivers
ACK	受信確認応答	receivers to a server
NACK	再送要求通知	receivers to a server
CONN	コネクション確立通知	a server to receivers
CACK	コネクション確立応答	receivers to a server
REL	コネクション開放通知	a server to receivers
RACK	コネクション開放応答	receivers to a server
POLL	調査通知	a server to a receiver
BUSY	BUSY 状態通知	a receiver to a server
RRDY	BUSY release	a receiver to a server
ABORT	受信失敗通知	a receiver to a server

表 C.1　RMTP の通信処理における主なパケット一覧

付録 D

通信ログのサンプル

```
Thu Sep 28 13:49:02 2000 RMTPD START (RMTP V2, NORMAL MODE, Infocast V2.5.1a)
DELIVERY START
Thu Sep 28 13:49:02 2000 send mode = normal memory mode (0)
Thu Sep 28 13:49:02 2000 send_file=/export/home/tao/contents/2/10m_1
Thu Sep 28 13:49:02 2000 file name=10m_1
Thu Sep 28 13:49:02 2000 sendkey=/var/tmp/eaa0U70p7
Thu Sep 28 13:49:02 2000 send_rate=200
Thu Sep 28 13:49:02 2000 unicast resending=1
Thu Sep 28 13:49:02 2000 scramble=0
Thu Sep 28 13:49:02 2000 auth=0
Thu Sep 28 13:49:02 2000 multicast addr=224.5.100.103
Thu Sep 28 13:49:02 2000 ttl=3
Thu Sep 28 13:49:02 2000 i/f=0.0.0.0
Thu Sep 28 13:49:02 2000 fsize=0
Thu Sep 28 13:49:02 2000 massblk=0
Thu Sep 28 13:49:02 2000 mbsize=0
Thu Sep 28 13:49:02 2000 per_nack=0
Thu Sep 28 13:49:02 2000 per_susp=0
Thu Sep 28 13:49:02 2000 send_irate=100
```

```
Thu Sep 28 13:49:02 2000 backoff=100

Thu Sep 28 13:49:02 2000 no_busy=0

Thu Sep 28 13:49:02 2000 no_nack=0

Thu Sep 28 13:49:02 2000 no_poll=0

Thu Sep 28 13:49:02 2000 port=7001

Thu Sep 28 13:49:02 2000 n_conn=1

Thu Sep 28 13:49:02 2000 n_poll=1

Thu Sep 28 13:49:02 2000 n_rack=1

Thu Sep 28 13:49:02 2000 n_rrdy=1

Thu Sep 28 13:49:02 2000 t_cack=100

Thu Sep 28 13:49:02 2000 t_acknack=100

Thu Sep 28 13:49:02 2000 t_rack=300

Thu Sep 28 13:49:02 2000 t_rrdy=50

Thu Sep 28 13:49:02 2000 width=0

Thu Sep 28 13:49:02 2000 mu_sel=0

Thu Sep 28 13:49:02 2000 uc_max=1

Thu Sep 28 13:49:02 2000 any=0

Thu Sep 28 13:49:02 2000 maxlc=5

Thu Sep 28 13:49:02 2000 maxrc=-1

Thu Sep 28 13:49:02 2000 comp=100

Thu Sep 28 13:49:02 2000 maxtime=-1

Thu Sep 28 13:49:02 2000 nabort=1

Thu Sep 28 13:49:02 2000 i_abort=100

Thu Sep 28 13:49:02 2000 nrel=1

Thu Sep 28 13:49:02 2000 i_rel=100

Thu Sep 28 13:49:02 2000 nmconn=0
```

```
Thu Sep 28 13:49:02 2000 i_mconn=0
Thu Sep 28 13:49:02 2000 SEND START
Thu Sep 28 13:49:02 2000 MULTICAST ADDR 224.5.100.103
Thu Sep 28 13:49:02 2000 file_size = 10240798
Thu Sep 28 13:49:02 2000 blk_no = 10001
Thu Sep 28 13:49:02 2000 mass_blk=0
Thu Sep 28 13:49:02 2000 cli_num = 1
Thu Sep 28 13:49:03 2000 type=512 len=2
Thu Sep 28 13:49:03 2000 type=513 len=4
Thu Sep 28 13:49:03 2000 type=514 len=9
Thu Sep 28 13:49:03 2000 time   1.390 sec CONNECTED 1
time  50.040 sec SENDING retry=0
DELIVERY END
RESULT START
210.162.187.241      pepper           COMPLETE    0 0
PacketType  Send       Receive
DT             10001         0
ACK                0         1
NACK               0         0
BUSY               0         0
RRDY               0         0
ABORT              0         0
POLL               0         0
CONN               1         0
CACK               0         2
REL                1         0
```

```
RACK                0         1

MOVE                0         0

SUSP                0         0

FREPO               0         0

time  63.900 sec SEND COMPLETE

CPU USER    0.140 sec SYSTEM    0.160 sec

RESULT END

Thu Sep 28 13:50:06 2000 SEND END
```

付録 E

CSV 変換プログラムソースコード

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define BUFLEN 256

void main(int argc, char *argv[])
{
    char buf[BUFLEN], *ptr ,*p;
    FILE *fpin,*fpout;
    int m,n=0; // MAX number of RETRY
    int n_cli=0; // MAX number of CLIENT

    int x=0;
    int i,s=0;

    /* File I/O */
    if((fpin=fopen(argv[1], "r"))==NULL)
    {
```

```
        printf("File Open Error.\nCommand : log2csv [LogFilename]
        [OutputFilename.csv]\n");
        exit(1);
    }
    if((fpout=fopen(argv[2], "w"))==NULL)
    {
    fclose(fpin);
        printf("File Write Error.\n");
        exit(1);
    }

/* Search MAX number of RETRY and CLIENT */
    do {
        ptr=fgets(buf, BUFLEN, fpin);

        /* Search MAX number of RETRY */
        p=strstr(buf,"retry");
        if (p!=NULL)
        {
            m+=1;
        }

/* Search MAX number of CLIENT */
p=strstr(buf,"cli_num");
if(p!=NULL)
{
```

```
if (p+10 > n_cli){
n_cli=p+10;
}
}

        p=strstr(buf,"RMTPD START");
        if (p!=NULL)
        {
            if(n<m) n=m;
            m=0;
        }
    } while( ptr != NULL );

    /* Show header contents */
    fprintf(fpout,
    "File,Rate,TotalSize,Blk_n,MbSize,BackOff,Start,FileSize,
                              Client,Connection,Transfer,");
    for(m=0; m<n-1; m++){ fprintf(fpout, "retry=%d,", m+1); }
    fprintf(fpout, "COMPLETE,FAIL,DT,ACK,NACK,POLL,
                             CONN,CACK,REL,RACK,Time,End\n");

    m=0;
    rewind(fpin); // fpin-pointer move head of file

    do {
        ptr=fgets(buf, BUFLEN, fpin); // Read a line
```

```
        /* file name */
        p=strstr(buf,"file name");
        if (p!=NULL)
        {
fputc( '\n', fpout ); // END_CHECK

            for (i=0; *(p+strlen("file name=")+i)!='\n'; i++){
                fputc( *(p+strlen("file name=")+i), fpout );
            }
            fputc( ',', fpout );
        }

        /* send_rate */
        p=strstr(buf,"send_rate");
        if (p!=NULL)
        {
            for (i=0; *(p+strlen("send_rate=")+i)!='\n'; i++){
                fputc( *(p+strlen("send_rate=")+i), fpout );
            }
            fputc( ',', fpout );
        }

        /* TotalSize */
        p=strstr(buf,"fsize");
        if (p!=NULL)
```

```
{
    for (i=0; *(p+strlen("fsize=")+i)!='\n'; i++){
        fputc( *(p+strlen("fsize=")+i), fpout );
    }
    fputc( ',', fpout );
}

/* massblk */
p=strstr(buf,"massblk");
if (p!=NULL)
{
    for (i=0; *(p+strlen("massblk=")+i)!='\n'; i++){
        fputc( *(p+strlen("massblk=")+i), fpout );
    }
    fputc( ',', fpout );
}

/* mbsize */
p=strstr(buf,"mbsize");
if (p!=NULL)
{
    for (i=0; *(p+strlen("mbsize=")+i)!='\n'; i++){
        fputc( *(p+strlen("mbsize=")+i), fpout );
    }
    fputc( ',', fpout );
}
```

```
/* buckoff */
p=strstr(buf,"backoff");
if (p!=NULL)
{
    for (i=0; *(p+strlen("backoff=")+i)!='\n'; i++){
        fputc( *(p+strlen("backoff=")+i), fpout );
    }
    fputc( ',', fpout );
}

/* SEND START */
p=strstr(buf,"SEND START");
if (p!=NULL && s==0)
{
    for (i=0; i<8 ; i++){
        fputc( buf[11+i], fpout );
    }
    fputc( ',', fpout );
    s++;
}
else if (p!=NULL)
{
    fputc( ',', fpout );
}
```

```
/* file_size */
p=strstr(buf,"file_size");
if (p!=NULL)
{
    for (i=0; *(p+strlen("file_size = ")+i)!='\n'; i++){
        fputc( *(p+strlen("file_size = ")+i), fpout );
    }
    fputc( ',', fpout );
}

/* cli_num */
p=strstr(buf,"cli_num");
if (p!=NULL)
{
    for (i=0; *(p+strlen("cli_num = ")+i)!='\n'; i++){
        fputc( *(p+strlen("cli_num = ")+i), fpout );
    }

    fputc( ',', fpout );
}

/* TimeForConnectionToClient */
p=strstr(buf,"CONNECTED");
if (p!=NULL)
{
    x++;
```

```
            for (i=0; i<7 ; i++){
                fputc( buf[30+i], fpout );
            }

            fputc( ',', fpout );
        }

        /* TimeForTransfer & TimeForRetry */
        p=strstr(buf,"retry");
        if (p!=NULL)
        {
            if(x!=1)
            {
                fputc( ',', fpout );
                x=1;
            }

            m++;
            for (i=0; i<7 ; i++){
                fputc( buf[5+i], fpout );
            }
                fputc( ',', fpout );
        }

/* COMPLETE/FAIL TARMINAL */
if ( !strncmp(buf+38, "COMPLETE", 8) )
```

```
{
for(i=0; i<10 ; I++){
fputc( buf[21+i], fpout );
}

fputc( ',', fpout );
}

if ( !strncmp(buf+38, "FAIL", 4) )
{
for(i=0; i<10 ; I++){
fputc( buf[21+i], fpout );
}
}

        /* DT */
        if( !strncmp(buf, "DT", 2) )
        {
            if(n!=m)
            {
                for (i=n-m; i!=0; i--){
                fputc( ',', fpout );
                }
            }

            for (i=0; i<5 ; i++){
```

```
        fputc( buf[14+i], fpout );
    }
    fputc( ',', fpout );
}

/* ACK */
if( !strncmp(buf, "ACK", 3) )
{
    for (i=0; i<2 ; i++){
        fputc( buf[28+i], fpout );
    }
    fputc( ',', fpout );
}

/* NACK */
if( !strncmp(buf, "NACK", 4) )
{
    for (i=0; i<2 ; i++){
        fputc( buf[28+i], fpout );
    }
    fputc( ',', fpout );

}

/* CONN */
if( !strncmp(buf, "CONN", 4) )
```

```
{
    for (i=0; i<2 ; i++){
        fputc( buf[18+i], fpout );
    }
fputc( ',', fpout );
}

/* CACK */
if( !strncmp(buf, "CACK", 4) )
{
for (i=0; i<2 ; i++){
        fputc( buf[28+i], fpout );
    }
    fputc( ',', fpout );
}

/* COMPLETE */
p=strstr(buf,"SEND COMPLETE");
if (p!=NULL)
{
    for (i=0; i<8; i++){
        fputc( buf[5+i], fpout );
    }
    fputc( ',', fpout );
}
```

```
        /* SEND END */
        p=strstr(buf,"SEND END");
        if (p!=NULL)
        {
            for (i=0; i<8; i++){
                fputc( buf[11+i], fpout );
            }
            fputc( ',', fpout );

            m=0;
            x=0;
            s=0;
    }

    } while( ptr != NULL );

    // printf("MAX number of RETRY = %d",n);

    fclose(fpin);
    fclose(fpout);
}
```