卒業論文 2005 年度 (平成 17 年度)

アドホックネットワークにおけるネットワーク負荷を考慮したコンテンツフラッディングモデルの提案

慶應義塾大学 環境情報学部
大薮 勇輝
t02171yo@sfc.keio.ac.jp

指導教員
徳田 英幸
村井 純
楠本 博之
中村 修
高汐 一紀
湧川 隆次

平成 18 年 2 月 15 日

卒業論文要旨 2005 年度 (平成 17 年度)

アドホックネットワークにおけるネットワーク負荷を考慮したコンテンツフラッディングモデルの提案

概要

近年，新しい情報インフラストラクチャとして，アドホックネットワークが社会的に注目されはじめている．アドホックネットワークは隣接する無線端末同士が直接もしくはマルチホップ通信を利用することにより，動的に構築されるネットワークである．これまでに，アドホックネットワークに関する研究活動が活発に行われることで，アドホックネットワークの利用可能性が高まっている．

一方で，アドホックネットワークは，局所的に構築されること，各ノードが移動することにより，非常に不安定なネットワークになっている．そのため，インターネットで安定して利用可能な通信方式が，アドホックネットワークでは安定して利用出来ないという問題がある．本研究では，アドホックネットワークの特徴と通信方式を比較することにより，フラッディングがアドホックネットワークに適した通信方式であることを示し，フラッディングを利用したアプリケーションモデルとして，広告配信モデルの提案を行った．

フラッディングはアドホックネットワークに適した通信方式であるが，一方で，過剰なネットワークトラフィックが発生するという問題がある．したがって，提案モデルを実現するためには，発生するネットワークトラフィックを抑制する必要がある．本研究では，フラッディングにより発生するネットワークトラフィックを抑制する手法として，Adaptive Application Flooding を提案した．

Adaptive Application Flooding は，(1) 中継ノードの選択機能，(2) データフロー数の削減機能，(3) データ間引き機能，(4) パケットサンプリング機能を利用することにより，自律的なネットワークトラフィックの抑制を実現する．本研究では，Adaptive Application Flooding の実装，動作実験を行うことによりこれらの機能を評価し，Adaptive Applicatoin Flooding の有意性を示した．

キーワード

1, アドホックネットワーク
2, コンテンツフラッディング
3, ネットワークトラフィック
4, データフローサイズの最適化

慶應義塾大学 環境情報学部

大薮 勇輝

Abstract of Bachelor's Thesis Academic Year 2005

An Adaptive Application Flooding for Efficient Data Dissemination in Ad-hoc Network

Summary

Mobile ad-hoc network is beginning to have gotten a lot of attention recently as a new communication infrastructure. Mobile ad-hoc network is a network established dynamically, by using direct or multi-hop communication between neighboring radio terminals. So far, the reality of mobile ad hoc network is increasing by the research activities about mobile ad hoc network being performed actively.

On the other hand, the ad hoc network is a very unstable network, because each node moves and being built locally. Therefore, there is a problem that the communication system which can be used stably on the Internet cannot be used stably on the mobile ad hoc network. In this research, we showed that a flooding is a transmission method suitable for mobile ad hoc network, by comparing the feature and other transmission methods within an ad hoc network. And we proposed the advertising distribution model as an application model using the flooding.

Although flooding is a transmission method suitable for an ad hoc network, it causes the huge amount of redundant traffic. Because of this, it is necessary to control and hold down the traffic in order to realize the proposal model. In this research, we proposed Adaptive Application Flooding as the technique of controlling the traffic generated by flooding.

Adaptive Application Flooding controls autonomously network traffic by using (1) the selection function of relay node, (2) the reduction function of the number of data flow, (3) the sampling processing function of packet sizes, and (4) the reduction function of the number of packet transmission per second. We evaluated these functions that Adaptive Application Flooding has, and showed the significance of Adaptive Applicatoin Flooding.

Key Word

1, mobile ad-hoc network
2, contents flooding
3, network congestion
4, data size adaptation

Keio University, Faculty of Environmental Information

Yuki Oyabu

目 次

図 目 次

表 目 次

第1章　序論

本章では本研究の背景及び目的を示す．まずアドホックネットワークの概要を示し，次にアドホックネットワークを普及させるために必要な事項を整理し，本研究の目的を示す．

1.1　アドホックネットワークの普及

PDA や携帯電話など無線端末の高性能化・小型化が実現されることにより，様々な場所で，移動しながらインターネットなどの情報インフラストラクチャを利用することが可能になった．無線端末の情報インフラストラクチャへの接続形態はインフラストラクチャモードとアドホックモードの 2 つに分類することが出来る．インフラストラクチャモードは無線端末間の通信がアクセスポイントやアクセスルータ経由で行われる接続形態であり，現在一般的に利用されている．一方，アドホックモードは，アクセスポイントなどのインフラストラクチャを必要とせずに，無線端末同士が直接通信を行う接続形態である．このアドホックモードを利用して構築されるマルチホップネットワークのことをアドホックネットワークという．

アドホックネットワークではエンドノード間の通信は直接もしくは複数の無線端末を経由して行われる．アドホックネットワークの大きな特徴は無線端末が移動しながら動的にフラットなネットワークを構築することにある．この特徴を利用することで，アクセスポイントなどのインフラストラクチャが無い場所においても無線端末同士を持ち寄るだけで局所的なネットワークを構築することが出来る．アドホックネットワークを利用することで災害時にアクセスポイントなどの固定されたインフラストラクチャが破壊された場合においてもすぐに代替ネットワークを構築することが出来る．

アドホックネットワークに関する研究活動はこれまでに多く行われており，ルーティングやグループコミュニケーションに関する研究が盛んに行われている．ルーティングに関しては，Reactive 型，Proactive 型，Hybrid 型 [1] という異なるルーティング方式が提案されている．Reactive 型ルーティングプロトコルとしては AODV[2]，Proactive 型のルーティングプロトコルとしては OLSR[3][4]，TBRPF[5] などが IETF において標準化された．グループコミュニケーションに関してはマルチキャスト [6][7] やフラッディング [8][9][10] の研究が盛んで，それぞれ IP レイヤで実現される手法からアプリケーションレイヤで実現される手法など，多くの手法が提案されている．また，ロケーションベースのプロトコル [11][12][13] など，アドホックネットワークの特徴を生かした手法の提案も多く行われている．

こうした研究活動の成果により，現在，アドホックネットワークを実社会において構築することは技術的に可能となった．しかし，アドホックネットワークの普及には，解決されるべき問題がまだ多く残されている．その中でも，アドホックネットワークにおける通信環境の改善，人々が日常的に利用可能かつユーザに利便性を提供するサービスアプリケーションの登場の 2 点はアドホックネットワークの普及にとって非常に重要な要素である．

1.1.1 通信環境の改善

アドホックネットワークの通信環境が不安定である理由は，無線通信の問題点及びアドホックネットワークの問題点から導かれる．無線通信の持つ問題点，アドホックネットワークの持つ問題点について以下に説明し，通信環境改善の必要性を示す．

無線通信の持つ問題点

無線通信の持つ問題点として以下の 3 点に注目する．これらは無線通信の問題点であると共に，アドホックネットワークにおける通信環境を悪化させる原因になる．

- 共有される通信帯域
  IEEE802.11 ワーキンググループで策定されている無線 LAN 規格で利用される周波数帯の電波は，同心円上の一定範囲に送信されるという特徴を持つ．そのため，送信するフレームは同心円上の一定範囲全ての通信帯域を消費する．仮に，ノード A が大量のフレームを送信しているとすると，その隣接ノードの通信帯域はノード A の送信するフレームにより，大量に消費されることになる．

- 電波干渉
  電波干渉は同じ通信帯域を利用する，異なる端末が全く同時にフレームを送出した場合に生じる．電波干渉の発生はフレームの衝突を意味しており，フレームの衝突が発生することにより，それぞれのフレームを構成する電波の波形に乱れが生じる．これにより，送信元と送信先の間でフレーム内容が異なってしまう．この結果，電波干渉はフレームの損失もしくは伝送遅延の原因となる．

- 伝送遅延
  無線 LAN 規格には，電波干渉によるフレーム損失を防止する機能として，CSMA/CA(搬送波感知多重アクセス/衝突回避方式) が備わっている．CSMA/CA とは無線インタフェースに受信されるフレームを監視することにより，その時点でフレームが送信されているか否かの検知を行い，フレームが送信されている場合は待ち，送信されていない場合はフレームを送信するアルゴリズムである．この CSMA/CA は大量のフレームが送信される環境においては，伝送遅延の原因となる．伝送遅延は CSMA/CA により，フレームの送信タイミングが遅れることにより生じる．伝送遅延により生じる問題としてはスループットの低下が挙げられる．

以上，無線通信の特徴として，共有される通信帯域，電波干渉，伝送遅延の 3 点に関して説明した．これら 3 点に関して可能な対策としては，より通信速度の高い無線規格の利用，CSMA/CA などデータリンクレイヤの各種プロトコルの改善・開発などが挙げられる．

アドホックネットワークの持つ問題点

アドホックネットワークの持つ問題点は，以下の特徴から生じる．

- フラットなネットワーク構造
  アドホックネットワークは，各ノードがマルチホップ通信を行うために，階層構造を持た

ない，フラットなネットワーク構造を持つといえる．無線通信の問題点を電波干渉，伝送遅延に関して説明したが，これらの問題はフラットなネットワークにおいて，より深刻な問題となる．このことを図 1.1に示す.

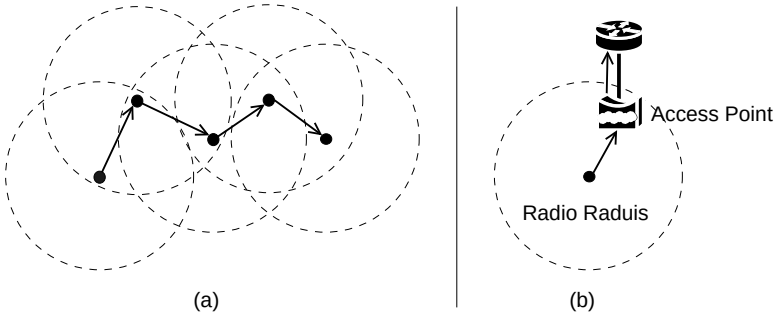

図 1.1: フラットなネットワーク構造におけるデータ転送

図 1.1(a) はフラットなネットワーク (アドホックネットワーク)，図 1.1(b) は階層構造を持つネットワーク (インターネット) を示している．(a) では，隣接ノードがパケットの中継を行うために，ノードの通信帯域が多く消費されている．一方で，(b) においてアクセスポイントはルータにパケットを転送するため，ノードの通信帯域はノード自身の送信した分だけ消費される．また無線範囲の重複に注目すると，フラットなネットワークにおいてはより多くの電波干渉，パケットロスが発生すること，さらに電波干渉により伝送遅延が発生することがわかる．このようにフラットなネットワークでは，通信帯域が多く消費される，電波干渉・伝送遅延が起こりやすいという問題がある．この問題を解決するためには，無線通信と同様に，データリンクレイヤにおける対策が必要である．

- 各ノードの移動性
  アドホックネットワークにおいてはトポロジ変化が頻繁に起こる．そのためアドホックネットワークでは，ある端末との通信を行っている際に途中経路が頻繁に消失する可能性がある．これはパケットロスやスループット低下の大きな要因となる．この問題への対処は主に経路修復もしくは冗長経路の確保などの方法が考えられている．経路修復，冗長経路の確保はともにネットワークレイヤでの対処方法が考えられており，前者に関してはルーティングプロトコルによる対応，後者に関しては新しいプロトコルの研究が行われている．

以上，アドホックネットワークにおいて通信環境が不安定になる要因として，無線通信の持つ問題点，アドホックネットワークの持つ問題点を示した．現在，これらの問題を解決する決定的な技術は提案されていない．したがって，アドホックネットワークにおける通信環境の改善には，より多くの研究成果が複数のレイヤに渡って上げられる必要がある．

1.1.2 サービスアプリケーションの必要性

アドホックネットワークの普及には，通信環境の改善を実現するだけではなく，アドホックネットワークで利用可能かつユーザに利便性を提供するサービスアプリケーションの提供が必要となる．インターネットが一般に普及した背景には，WEB やチャット，メールといったサー

ビスの登場，そしてこれらのサービスを利用するためのアプリケーションの充実が挙げられる．これらのサービスは日常生活に必要な情報の取得が可能，遠く離れた相手とのコミュニケーションが可能といった利便性を人々に提供することに成功している．

しかし，アドホックネットワークは特に災害時における利用方法が多く提案されているが，日常生活において利用可能な，利便性の高いサービスモデルの提案はあまり行われていない．インターネットが日常生活に利便性の高いサービスの提供により普及したことを考慮すると，災害時など特別な状況でしか主に利用されないネットワークは一般には普及しない．このように，アドホックネットワークが一般に普及するためには，人々が日常的に利用可能であり，利益を得られるようなサービスアプリケーションがアドホックネットワークにおいて展開される必要がある．

1.2 本研究の目的

本研究は，1.1節で述べたように，通信環境の改善とサービスアプリケーションの提案の 2 点を研究対象とする．また，アドホックネットワークにはインターネットへの接続性を提供するモデル，接続性を持たないモデルの 2 種類あるが，本研究では後者を扱う．本研究では，アドホックネットワークにおいて，ユーザに利便性を提供するサービスモデルの提案を行い，提案モデルを，無線端末の密集する場所において，安定して利用するための要件を整理する．さらにまとめた要件に基づき，提案モデルを利用する全てのアプリケーションが利用するべきアプリケーション基盤を構築する．本研究の目的は，構築したアプリケーション基盤の評価を行うことにより，提案モデルの実現可能性を示すことにある．

1.3 論文の構成

本論文は全 9章から構成される．第 2章では，アドホックネットワークの特徴と通信方式の特徴の比較を基に，日常的に利用可能かつユーザに利便性を提供するサービスモデルの提案を行う．第 3章では，提案モデルを利用する際に生じる問題の提起・整理を行い，第 4章において提起した問題に対する既存のアプローチを示し，本研究の取るべきアプローチに関して考察を行う．第 5章では，問題点に対し本研究の取るアプローチを提案し，第 6，7章ではそれぞれ，構築するアプリケーション基盤の設計，実装を示す．第 8章では，構築したアプリケーション基盤の評価を行い，結論を第 9章にまとめる．

第2章　アドホックネットワークに適したサービスモデルの提案

本章では，日常生活において利用可能な，人々に利益をもたらすサービスモデルを提案する．まず，アドホックネットワークに適した通信方式，アプリケーションに関して考察する．次に，これらの考察を基に本研究の対象とするサービスモデルを提案する．

2.1　アドホックネットワークに適した通信モデルの考察

第1章で示したように，アドホックネットワークはインターネットに比べ，不安定な通信環境を持つネットワークである．したがって，アドホックネットワーク上で利用されるアプリケーションはアドホックネットワークの持つ特徴を考慮して構築される必要がある．本節では通信モデルや，通信モデルに対応する通信方式について考察する．その後，アドホックネットワークの特徴に適した通信方式について議論する．

2.1.1　アプリケーションの利用する通信モデル

アプリケーションの利用する通信モデルは大きく 1 対 1 通信と 1 対特定多数通信，1 対不特定多数通信の 3 種類に分類される．

1 対 1 通信モデル

まず 1 対 1 通信モデルがどのような特徴を持つか考察する．1 対 1 の通信モデルは，インターネットにおいて，多くのアプリケーションから利用されている通信モデルである．例えば，WEB やメール，メッセンジャーなど，一般的に利用されるアプリケーションのほとんどが 1 対 1 通信を利用している．1 対 1 通信の特徴は通信相手を明示的に選択するか，通信相手として選択されることによって通信が開始されることである．また，1 対 1 通信はトランスポート機能として，コネクション型の TCP，コネクションレス型の UDP が用いられる．

TCP は WEB やファイル交換ソフトなど多くのアプリケーションから利用されている．それは，TCP を利用することにより通信の信頼性が保証されるためである．通信の信頼性とは，送信したパケットが必ず届くこと，送信したパケットが送信先で順番通りに処理されることを指す．また，TCP は信頼性を提供するため，規模性を実現するための機能として輻輳制御を行う．輻輳制御は，送信先までのネットワーク環境に異常を検知することにより，転送速度を調整する．この際，TCP はネットワーク環境が改善されるまで，低い転送速度で通信するように動作する．これにより，TCP はネットワークトラフィックの集中するネットワークにおいても，自律的に転送速度を調整し，ネットワーク環境に応じたデータ転送を実現することが出来る．

一方，UDP は TCP のように輻輳制御を行わず，信頼性も保証されない．UDP はその代わ

りに，ネットワーク環境の変化に依存せず，一定の転送速度で送信先にデータを送信することが出来る．そのため，リアルタイムなストリーミングなど，一定の通信帯域を常時要求するアプリケーションから利用される．

1 対特定多数通信モデル

1 対特定多数通信モデルを利用する通信方式としてマルチキャストが挙げられる．マルチキャストは，特定の複数ノードに対して情報を同時に送信する通信方式である．マルチキャストは，主に ISP の持つネットワーク内に限定された情報配信サービス (映像等) に利用されているものの，インターネット規模では利用されていない．マルチキャストの特徴は通信相手の決定に際し，通信相手のアドレスではなく，マルチキャストアドレスを利用することである．一方，マルチキャストを利用して送信されたデータを受信するためには，ノードから join リクエスト (受信要求) が送信される必要がある．join リクエストの送信により，ノードはそのマルチキャストデータが送信されるマルチキャストグループに参加することが出来る．また，通常のマルチキャストでは輻輳制御は行われないが，階層符号化マルチキャスト [14][15] など輻輳制御を実現する手法が複数提案されている．

1 対不特定多数通信モデル

1 対不特定多数通信モデルは現在一般的に利用されていない．1 対不特定多数通信モデルを利用する通信方式としてはフラッディングが挙げられる．フラッディングを利用することにより，一度に不特定のノードに対し情報伝達が可能なものの，その情報を受け取ったルータ全てが再送を行うために，通信帯域に及ぼす影響が他の通信モデルに比べ高い．フラッディングの特徴は通信相手の指定や，経路情報を必要とせずに情報伝搬が出来ることである．また，通常のフラッディングにおいては輻輳制御は利用されない．

通信方式の比較

各通信モデルに関する特徴，各通信モデルに対応する通信方式に関して特徴を述べた．表 2.1には各通信モデルの特徴をまとめた．マルチキャストに関しては階層符号化マルチキャストの利用を考慮するため，三角をつけた．また，通信相手の指定を行う通信モデルではデータ送信に経路情報を利用することを示している．

表 2.1: 各通信モデルの特徴

通信モデル	通信方式	通信相手の指定	輻輳制御	利用頻度
1 対 1	TCP	受信ノードのアドレス	○	◎
	UDP	受信ノードのアドレス	△	○
1 対特定多数	マルチキャスト	マルチキャストアドレス	△	△
1 対不特定多数	フラッディング	不要	×	×

2.1.2 アドホックネットワークの特徴が各通信方式に与える制約

本節ではアドホックネットワークの特徴に注目し，それらの特徴が表 2.1にまとめた通信方式に対して，どのような影響を与えるか示す.

各ノードの移動によるトポロジ変化

まず，アドホックネットワークが動的に構築されるネットワークであることに注目する．通信相手までの通信に経路情報を利用するか否かが大きな違いとなる．アドホックネットワークのトポロジは各ノードの移動により頻繁に変化する．そのため，通信の最中にトポロジが変化する可能性が非常に高い．アドホックネットワークにおけるトポロジ変化を図 2.1に示す.

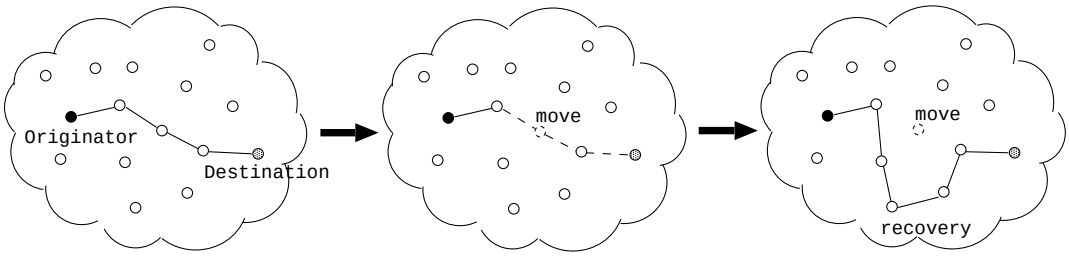

図 2.1: アドホックネットワークにおけるトポロジ変化

図 2.1に示すように，アドホックネットワークでは通信相手までの経路が消失した場合に，自律的に経路の修復が行われる．この経路修復はルーティングプロトコルにおける HELLO メッセージの送出時間によるものの，数秒から数十秒の時間が必要となる.

ここで，経路修復が各通信方式に及ぼす影響を考察する．まず，TCP の場合は輻輳制御が行われることにより転送速度の調整が行われる．さらに，経路修復にかかる時間が数秒であることを考慮すると，転送速度は大幅に調整される．また，頻繁に経路損失が起こる場合には，輻輳制御が頻繁に行われることにより低い転送速度での通信が続くことになる．次に，UDP 及びマルチキャストの場合を考察する．これらの通信方式では通常輻輳制御は行われない．そのため，経路損失による転送速度の低下はない．しかし，経路修復が行われるまでの間，損失した経路を利用するパケット全てが損失してしまう．最後にフラッディングに関して考察する．フラッディングの場合も輻輳制御が行われないが，フラッディングは経路情報を利用しない通信方式であるためトポロジ変化による経路消失などの影響を受けない.

アドホックネットワークの持つ局所性

次に，アドホックネットワークが局所的に構築されるネットワークであることに注目する．この特徴のために，アドホックネットワークにおいて利用されるアプリケーションは，限定された範囲内でのみ利用される．また，限定された範囲における通信では，通信相手を指定するか否かが大きな違いとなる.

通信相手を指定する場合の必要事項は，同じネットワークに通信相手が存在すること，通信相手に対応したアドレス，名前等 (FQDN) の情報がわかることの 2 点である．前者に関して考察すると，範囲の限定されたネットワークにおいて，指定した通信相手が同じネットワークに

参加している機会は少ない．また，後者に関してはDNS等の名前解決システムが必要となる．インターネットの場合は，名前解決機構として，DNSサーバへの到達性を常に確保し，安定した名前解決が必要である．

一方，アドホックネットワークではネットワークの局所性により，ネットワーク全体に配置されるノードの名前を一括して扱う名前解決機構を構築することが出来ない．そのため，一つのアドホックネットワーク上に複数の名前解決機構が必要になる．この場合，名前解決機構はその近隣に配置されるノードのみの名前解決を行うことになる．これは，インターネットにおける名前解決機構がインターネット全体の名前解決が可能であるのに対し，アドホックネットワークにおける名前解決機構では非常に限られた範囲に対してのみしか名前解決が出来ないことを意味する．以上を考慮すると，指定した相手の名前等が利用する名前解決機構の持つ情報群に含まれる可能性は非常に低い．さらに各ノードの移動によりトポロジ変化が発生した場合を考慮すると，名前解決機構までの経路損失が頻繁に発生する，名前解決に時間がかかるなど様々な問題が発生する．アドホックネットワークにおける名前解決機構は，これまでにいくつか提案されているが，上述した問題を決定的に解決するための手法はまだ研究段階であり，一般に利用されていない．

以上の理由から通信相手の指定が必要とされる通信方式の利用機会は少ない．一方，通信相手を指定しない場合の必要事項は，同じネットワークに通信相手が存在することのみである．通信相手を指定しない場合は，名前解決機構を必要としないため，通信相手を指定する場合に比べ，通信の機会が多い．

2.1.3 本研究の注目する通信方式

2.1.2節では，ノードの移動性によるトポロジ変化及びアドホックネットワークの局所性が各通信モデル，通信方式にどのように影響するか議論した．ノードの移動性に関しては，経路情報の冗長化や代替経路を利用することにより，今後解決される余地があるが，現時点においてトポロジ変化に対する経路修復までには数秒の時間がかかる．ノードの移動性に関しては，指定する通信相手が物理的にアドホックネットワークに参加しているかの問題であり改善することは出来ない．そのため，アドホックネットワークでは，通信相手を指定しないサービスアプリケーションの方が汎用的に利用可能である．

一方でフラッディングを利用する場合は，通信相手の指定，経路情報を必要としない．フラッディングは，インターネットにおいてほとんど利用されていない技術であるが，アドホックネットワークにおいて利用される場合には注目すべき特徴を持つ．この特徴を次に説明する．

アドホックネットワークにおけるフラッディングの特徴

アドホックネットワークは無線通信のみで構築されるフラットなネットワークであり，情報は無線端末を複数経由することにより伝達される．したがって，アドホックネットワークにおけるフラッディングは，情報の伝搬範囲が送信場所を中心に拡大するという特徴を持つ．この特徴を利用することで，送信ノードは距離的に近いノードに対して情報を伝達することができるため，送信場所に関係する情報を効果のある範囲に伝達することが出来る．

本研究で扱う通信方式

以上説明したように，アドホックネットワークにおいては通信相手の指定，経路情報を必要とないフラッディングが適している．さらに，アドホックネットワークにおけるフラッディングは送信場所に関連した情報を効果のある範囲に伝達できるという利点を持つ．以上の理由により，本研究ではフラッディングに注目する．本研究では以降，フラッディングを利用したアプリケーションモデルを提案する．

2.2 アプリケーションモデルの提案

2.1節での議論を踏まえ，本節ではフラッディングを利用したアプリケーションモデルの提案を行う．フラッディングを利用することで展開するべきサービスを考察すると，まず 1 対多通信を行うという性質上，多くの人々にとって有益な情報が送信されるべきである．また，ネットワークの局所性を考慮すると，限定された範囲内において価値のある情報である必要がある．これらの要求を満たす一つの例として，本研究ではフラッディングを利用した広告配信を提案する．本研究の想定する広告配信モデルを図 2.2に示す．

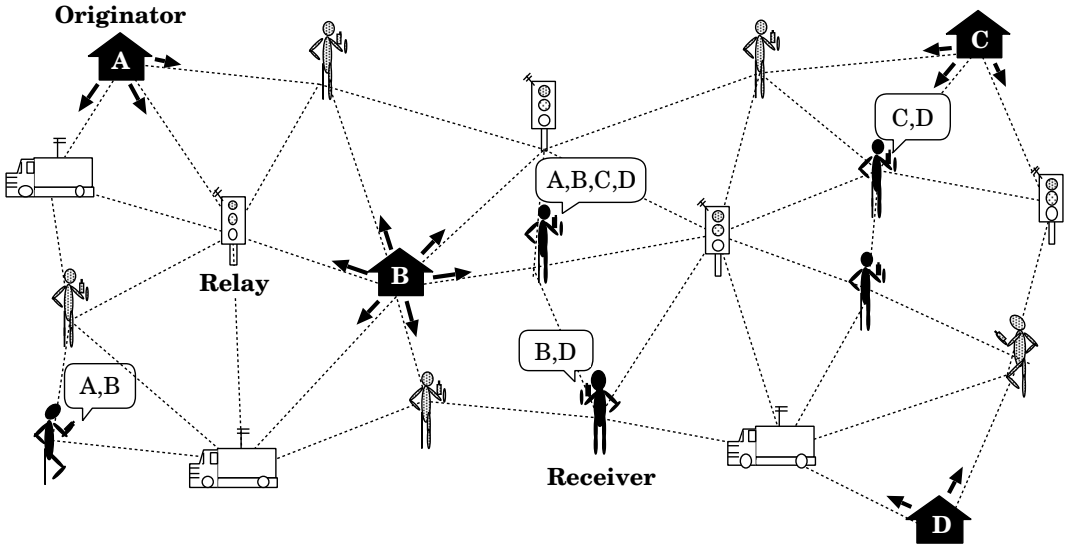

図 2.2: 提案するサービスモデル

本研究の提案する広告配信モデルは，図 2.2に示すように，街中に存在するショッピングモール，レストランや映画館など様々な建物，店に設置された送信ノードがそれぞれの広告をフラッディングしている環境を想定する．また，これらの送信ノードが送信する広告には，バーゲン情報，お勧めの音楽，放映されている映画の予告編など，様々な内容のものを想定する．図 2.2中に示した矢印は広告コンテンツの伝搬していく経路を示したものである．想定アプリケーションモデルでは，中継ノードが受け取ったデータを再度ネットワークに向けてフラッディングすることで，送信ノードを中心とした一定範囲内に広告を伝搬する．また，受信ノードはこれらの広告を移動しながら自動的に受け取る．以降本研究の想定シナリオにおけるノード種類とその役割，及び想定アプリケーションモデルが日常生活にもたらす利便性を説明する．

2.2.1 ノード種類別の役割

アドホックネットワークに参加するノードの多くがPDAや携帯電話など，演算能力の低い，バッテリー容量に限りのある小型端末であることを考慮すると，フラッディングを利用する場合には，各ノードの演算能力，バッテリー容量を考慮することが重要となる．なぜならフラッディングが通信方式として利用された場合は，中継ノードに対し大量の計算機資源，バッテリー消耗が要求されるからである．提案シナリオにおいてノードの種類毎に求められる事項を以下にまとめる．

- 送信ノード
  送信ノードは作成した広告データをネットワークにフラッディングする端末を指す．送信ノードはレストランやデパート，映画館など，公共の場所に設置される．送信ノードは広告データの作成・送信等を行うため，演算能力の高い，バッテリー容量の心配がない端末が想定される．送信ノードには隣接する出来る限り多くの端末に広告データを送信することが求められる．

- 中継ノード
  中継ノードは隣接する送信ノード，または中継ノードから受信した広告データの再送を行う端末を指す．中継ノードには受信ノードに比べ，高い演算能力，多くのバッテリー消耗が求められる．また，広告データをネットワークに伝搬するため，非常に重要な役割を持つ．したがって中継ノードには移動ノード以外にも信号機や中継車など演算能力，バッテリー容量が共に高い端末が利用されることが望ましい．

- 受信ノード
  受信ノードは送信ノードまたは中継ノードから広告データを受信する端末を指す．受信ノードにはPDAや携帯電話など，演算能力及びバッテリー容量が比較的少ない端末を想定する．受信ノードには受信した広告データを再生するためのスピーカーやディスプレイが付属されていることが望まれる．受信ノードは送信ノードのコンテンツ伝搬範囲に位置することで，自動的に広告データを受信する．

2.2.2 提案アプリケーションモデルの利点

本項では提案アプリケーションモデルのもたらす利点を示す．本研究の提案したアプリケーションモデルの利点は，次の2点である．

- 送信場所を中心とする広告伝搬範囲の拡大
- 広告情報の自動的な受信

1点目に関しては，広告場所を中心として広告データを伝搬することで高い広告効果が得られるという利益を送信者にもたらし，受信者に対しては受信者の場所に応じて周辺の店やイベントなどの情報が取得できるという利益をもたらす．2点目に関しては，受信者にとって，今まで全く知らなかった情報を取得できるという利点をもたらす．例えば，初めて訪れる地域でレストランを探しているときや，ショッピングをしているときなど，受動的に情報を取得できることは大きな利益となる．このように本研究の提案するアプリケーションモデルは日常生活に対して多くの利益をもたらすことが出来る．また，こうした広告配信はインターネットなど世

界規模のネットワークでは実現することが難しい．提案するアプリケーションモデルが展開されることで，より多くの人々がアドホックネットワークを利用する機会が増えることが考察される．

2.3 本章のまとめ

本章は，アドホックネットワークに適したサービスモデルを提案するために，各通信モデル及び通信方式の特徴に関してまとめ，それらの特徴がアドホックネットワークにおいてどのような影響を受けるか議論を行った．これらの議論の結果，アドホックネットワークに適した通信方式がフラッディングであることを示し，フラッディングを利用したサービスモデルとして広告配信モデルを提案した．さらに広告配信モデルがユーザに提供する利便性に関しても議論を行い，これにより，提案モデルがユーザに利益を提供するモデルであることを示した．本研究では以降，提案モデルを実現するための要件を示す．

第3章　本研究の問題点及びアプローチ

第2章にて提案したアプリケーションモデルが多くの利益をもたらすことを示した．しかし，本研究の提案したアプリケーションモデルには考慮すべき問題点がある．それはフラッディングを利用することによる，ネットワークトラフィックの増加である．本章ではまず，ネットワークトラフィックの増加により生じる問題点を示す．その後，過度のネットワークトラフィック発生の要因を示し，これに対しどのようなアプローチが可能か議論を行う．

3.1　ネットワークトラフィックの増加により生じる問題点

第1章にて，無線通信の問題点として共有される通信帯域，電波干渉，伝送遅延，3点を示した．本節では，ネットワークトラフィックの増加が無線通信に与える影響及び問題点を，これら3点に沿って示す．

まず，共有される通信帯域に関して考察する．ネットワークトラフィックの増加がより多くの通信帯域を消費することは明らかであるが，無線通信の場合には，あるノードAの通信により発生するネットワークトラフィックが，ノードAの隣接ノードの通信帯域を消費することになる．また802.11b/gなど，無線デバイスの利用可能帯域が理論値で11Mbps/54Mbpsであることを考慮すると，ネットワークトラフィックの増加は通信帯域の急激な消耗に繋がることがわかる．

次に電波干渉に関して考察する．アドホックネットワークのようにマルチホップな通信環境では，ネットワークトラフィックの増加は，複数の無線範囲の重なる地点において，大量の電波干渉，フレーム衝突の原因となる．そのため，ネットワークトラフィックが増大するにつれて，パケットロス率が増加することになる．

最後に，伝送遅延に関して考察する．伝送遅延はCSMA/CAの利用により生じる．ネットワークトラフィックの増加に伴い，CSMA/CAはフレームの衝突を回避するという点において，非常に重要な役割を持つが，アドホックネットワークのように無線範囲が複数重なる環境においては，送信するタイミングが得られず，伝送遅延が生じてしまい，結果としてスループットを低下させることになる．また，過度の伝送遅延が起こる場合は，デバイスのキューが詰まり，パケットロスが発生する．

以上示したように，ネットワークトラフィックの増大は，無線通信環境において，パケットロス率の向上，大量の伝送遅延を引き起こす．以降ではフラッディング利用時に生じるネットワーク負荷を示す．

3.2　フラッディングによるネットワークトラフィックの特徴及び問題点

フラッディングは，全ノードがパケットの再送を行うという性質上，大量のネットワークトラフィックを発生させる．そのため，フラッディングを利用する場合，無線の利用可能帯域に

対して過剰なネットワークトラフィックが発生する可能性が他の通信方式に比べ高い．そのため，2節で提案したサービスモデルを実現するためには，発生するネットワークトラフィックの抑制が必要となる．本節ではフラッディング利用時に発生するネットワークトラフィックの特徴を考察する．

3.2.1 フラッディング利用時の前提事項

フラッディングを利用する場合，送信・中継ノードの送信するパケットは，それを受信した隣接ノードにより再送される．そのため送信・中継ノードは隣接ノードから既に送信したパケットを重複して受信することになる．このパケットを送信・中継ノードが再送すると，パケットのループが発生する．このことを図 3.1を利用して説明する．

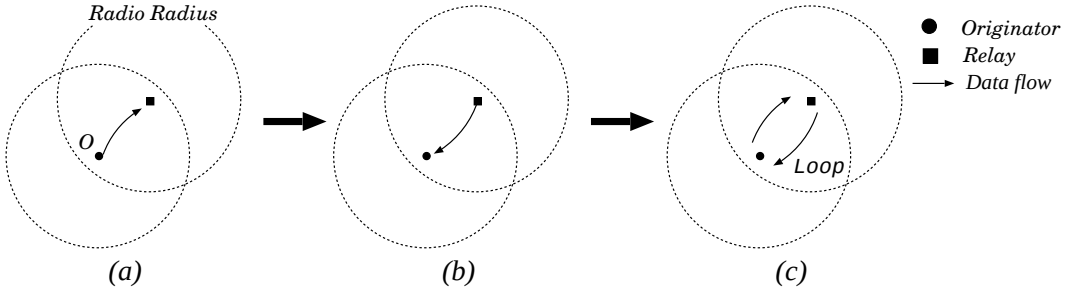

図 3.1: 重複パケット検知を行わない場合に発生するパケットのループ

図 3.1(a)(b)(c) はパケットのループがどのように発生するかを示す．(a) は送信ノードがパケットを送信する場合，(b) は中継ノードがそのパケットを再送する場合，(c) は中継ノードにより再送されたパケットを送信ノードがさらに再送する場合を示す．図 3.1(a)(b)(c) が示すように，送信ノードは既に送信したパケットと同一のパケットを再送するとパケットのループが発生する．このループを防ぐためには，全ノードが重複パケットの検知機能を備える必要がある．この機能は，既に送ったパケットを検知し，再送しないようにする機能である．重複パケットの検知機能は通常，IP ヘッダの拡張フィールドやパケットのデータヘッダに識別子を記述することで行われる．本研究では以降，フラッディングを利用する全ノードに重複パケットの検知機能が備わることを前提として，フラッディング利用時に発生するネットワークトラフィックの基本性質，本研究における定義を示す．

3.2.2 フラッディングによるネットワークトラフィックの基本性質

本項ではフラッディング利用時に発生するネットワークトラフィックの基本性質を示す．フラッディングされたパケットが送信・中継ノードに及ぼすネットワークトラフィックについて，図 3.2を用いて説明する．

図 3.2(a)(b) は，フラッディングされるパケットに注目し，そのパケットが送信・中継ノードに何度受信されるかを示している．図 3.2(a) は送信ノード O，図 3.2(b) は中継ノード R に注目している．実線で描かれた大きな円は無線範囲を示す．また，送信ノード O，中継ノード R の周りにはそれぞれ 4 つの中継ノードが存在することを仮定している．まず図 3.2(a) に注目す

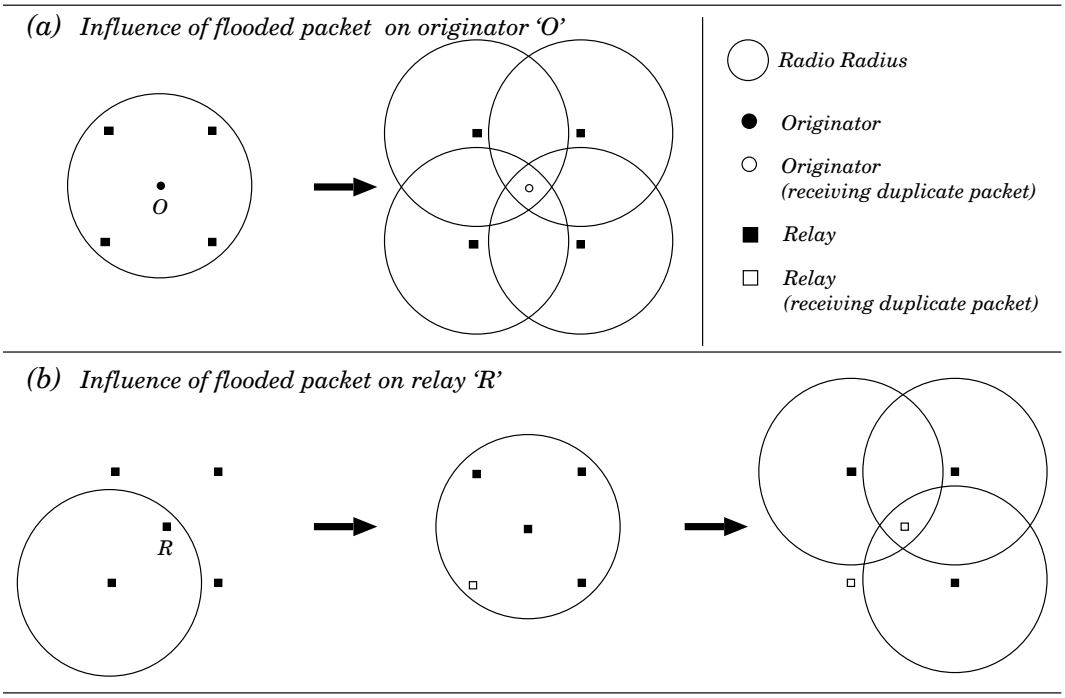

図 3.2: フラッディング利用時に生じるネットワーク負荷

ると，O の送信したパケットが，次の段階では，O に隣接する 4 ノードにより再送されていることがわかる．これは，送信ノードが，パケットを送信した直後に，隣接ノードと同じ数だけ同一内容のパケットを受信することを示している．次に図 3.2(b) に注目すると，R は隣接ノードからパケットを受信した後，受信したパケットを再送する．再送されたパケットは次の段階で，そのパケットを重複なく受信した R の隣接する 3 ノードにより再送される．したがって，中継ノードにおいても，送信ノードと同様に，隣接する中継ノード数と同数の，同一内容のパケットが受信される．このように，フラッディング利用時には，各ノードが同一内容のパケットを隣接ノード数に応じて受信することになる．以上，フラッディングの特徴を述べた．以降では，本研究におけるネットワークトラフィックの定義及び前提について説明した後，以上述べた特徴を踏まえ，フラッディングによるネットワークトラフィックの定義を示す．

本研究におけるネットワークトラフィックの定義及び前提

本研究では，ネットワークに送信されるデータフローにより消費される通信帯域のことをネットワークトラフィックとする．本研究では以降，フラッディングにより発生するネットワークトラフィックの定義を行う．以降では，定義を行う上で前提とする 2 つの事項を説明する．

- 通信帯域が変化しないこと
- 各ノードの無線範囲が同じ大きさであること

1 点目の前提を説明する．IEEE802.11a/b/g などの無線デバイスでは，通信品質として SNR(信号対雑音比) の値を持ち，SNR の示す値によって通信帯域を変化させる機能を持つ．この機能

を有効にすることにより，SNR が低下した場合にも，フレームの衝突を減少することが出来るが，通信帯域を変化させるための閾値及び動作は無線デバイスに依存する．これらの依存性を考慮すると，フラッディングによるネットワークトラフィックの定義が困難である．従って，本研究では，この機能を各ノードが利用しないこと，通信帯域が変化しないことを前提とする．

次に，2 点目の前提を説明する．実際の無線環境において，無線範囲は無線デバイス毎に異なる．そのため，実際の無線環境においてはノード A の電波はノード B に届くが，ノード B からノード A に対しては電波が届かない，といった現象が起こる．これらを考慮すると，フラッディングによるネットワークトラフィックの定義が困難である．従って，各ノード間がシンメトリックリンクであることを前提とする．無線端末が一定であることを前提とする．

以上，本研究におけるネットワークトラフィックの定義及び前提を示した．以降ではこれらの前提の下，フラッディングによるネットワークトラフィックの定義を行う．

ノード密度に起因するネットワークトラフィックの定義

フラッディングによるネットワークトラフィックを定義するために，まず隣接ノード数に注目する．隣接ノード数に注目すると，隣接ノード数と重複パケットの受信回数は正の相関関係を持つことがわかる．ここで，送信ノード a の送信するパケットのサイズを x_a，パケット送信回数/秒を p_a，送信ノード a の伝搬範囲に配置されるノード b に隣接する中継ノード数を N_b とすると，送信ノード a のフラッディングするデータフローによりノード b に対して発生するネットワークトラフィック T_a を示すことが出来る．

$$T_a \geqq (N_b + 1)x_a p_a \tag{3.1}$$

式 3.1は隣接する中継ノード数 N_b に加え，送信ノード a の送信するパケットのサイズ x_a，パケット送信回数/秒 p_a が通信帯域の消費量と正の相関関係を持つことを示している．また，式 3.1の右辺と左辺の関係が等しい場合は理想値であり，実際の無線環境を考慮すると左辺が右辺よりも大きい場合が考察される．式 3.1を利用することにより，送信ノード a の隣接ノード数 Nが 4，データフローのサイズ $x_a p_a$ が 100Kbps の場合に，500Kbps 以上のネットワークトラフィック T_a が発生することがわかる．

複数データフローに起因するネットワークトラフィックの定義

次にデータフロー数に注目する．式 3.1は単一のデータフローにより発生するネットワークトラフィックを示すが，複数のデータフローが送信される環境を想定すると，各ノードに対し発生するネットワークトラフィック T は，データフロー数を n，各データフローのデータサイズを $x_k p_k$ とおくと，式 3.2のように表せる．

$$T \geqq (N_b + 1)\sum_{k=1}^{n} x_k p_k \tag{3.2}$$

式 3.2により，複数のフローが送信される環境において，ノード b に対し発生するネットワー

クトラフィック Tが隣接する中継ノード数 N_b，データフロー数 n，各データフローのデータサイズ $x_k p_k$，それぞれと正の相関関係をもつことがわかる．

以上，フラッディング利用時に発生するネットワークトラフィックの定義を示した．本章では以降，ノード密度に起因するネットワークトラフィック，複数データフローに起因するネットワークトラフィック，それぞれに関して説明する．

3.3 ノード密度に起因するネットワークトラフィックの問題点

ノード密度に起因するネットワークトラフィックは，ノード密度の増加につれて，急激に増大するという特徴を持つ．また，ノード密度の高い環境においては，伝搬範囲の拡大に対して冗長なパケットが再送されることにより，不必要なネットワークトラフィックが発生するという問題がある．本節ではこれらの特徴，問題点を説明する．

3.3.1 ノード密度の増加により増大するネットワークトラフィック

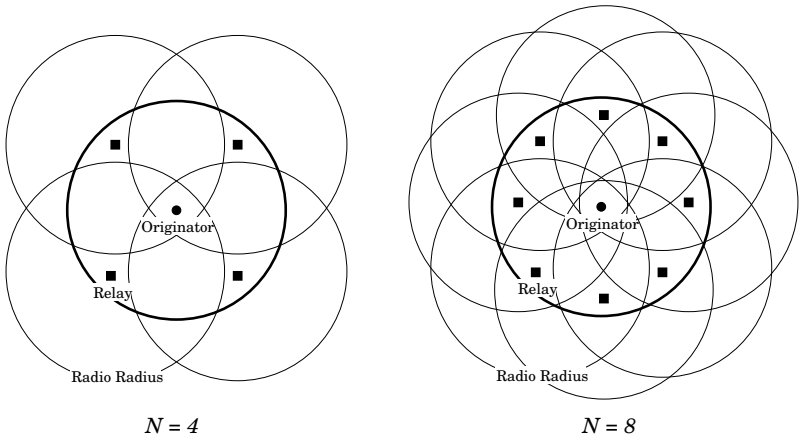

図 3.3: ノード密度の増加につれて増大するネットワークトラフィック

まず，ノード密度の増加につれて増大するネットワークトラフィックに関して説明する．式 3.1，図 3.3に示すように，通信帯域の消費量と隣接する中継ノード数は正の相関関係にある．このためノード密度が高くなるにつれて，フラッディングされるデータフローは各ノードの通信帯域をより消費することになる．また 802.11b などの無線デバイスの通信範囲が 100m 程度であることを考慮すると，ノード密度が高い環境において，隣接ノードが数 10 台にまで及ぶ可能性がある．隣接ノードが 50 台いる環境で 100Kbps のデータフローのフラッディングされることを想定すると，そのデータフローにより常時 5Mbps 以上の通信帯域が消費されることになる．このように，ノード密度の高い環境におけるフラッディングは，隣接ノード数が多くなるにつれ，非常に大量の通信帯域を必要とする．さらに 802.11a/b/g などの無線デバイスの利用可能帯域の理論値は 11/54Mbps と限られていることを考慮すると，必要とされる通信帯域が利用可能帯域を越える可能性が高い．

3.3.2 ノード密度の増加による冗長パケット数の増加

次に，コンテンツ伝搬範囲の拡大に関係のない冗長パケットの増加によるネットワークトラフィックの発生に関して説明する．フラッディングは伝搬範囲に配置される全ノードにパケットを伝搬するために利用される．しかしノード密度の高い環境においてフラッディングを利用する場合，伝搬範囲内の全ノードにパケットを伝達するために必須となるパケット以外に，冗長なパケットが送信される可能性が高い．そのためこの問題は先に説明した問題と密接に関わる．冗長パケットにより発生するネットワークトラフィックを図 3.4を用いて説明する．

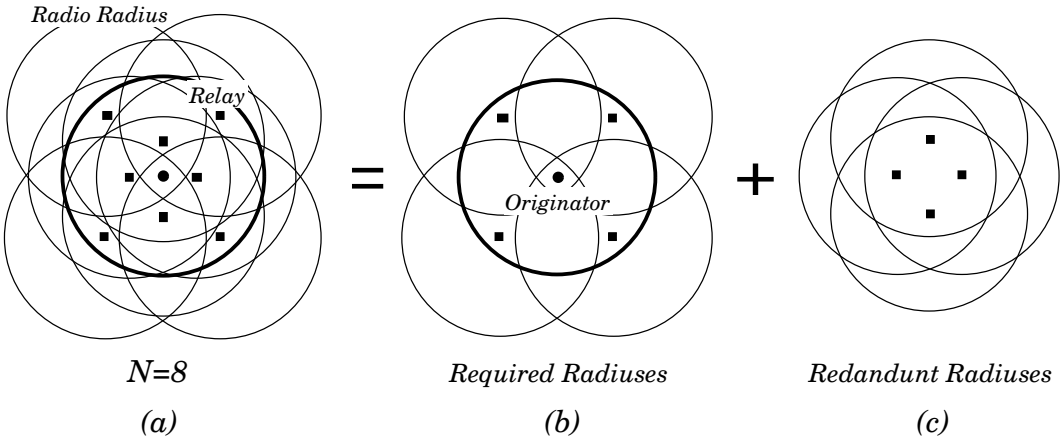

図 3.4: 冗長パケットにより発生するネットワークトラフィック

図 3.4(a)(b)(c) は 8 台の隣接ノードにより，フラッディングされたパケットの再送がされていく様子を示す．実線で描かれた円は各ノードの送信するパケットの到達範囲である．図 3.4(a) は送信ノードとその隣接ノードの無線範囲を示し，図 3.4(b)(c) はそれぞれ，伝搬範囲全体にパケットを伝搬する目的とって必須な無線範囲，冗長な無線範囲を示している．図 3.4(a) では，8 台いる隣接ノードのうち，4 台の隣接ノードの無線範囲が他のノードの無線範囲に含まれていることがわかる．この際，この 4 台の隣接ノードの再送するパケットにより，隣接ノードは重複パケットを無駄に多く受信することになる．また，図 3.4において 100Kbps のデータフローがフラッディングされたとすると，伝搬範囲全体にパケットを伝達するために必須となるパケットが消費する通信帯域は 500Kbps 以上，冗長となるパケットの消費する通信帯域は 400Kbps 以上となる． このように，隣接ノードの配置によっては，隣接ノードに冗長ノードが多く含まれるようになる．また，各ノードに対する冗長ノードの割合はノード密度が高くなるにつれて増加することがわかる．本研究では以降，パケットを伝搬範囲全体に伝達するために再送を求められるノードを必須ノード，再送の必要がないノードを冗長ノードと呼ぶ．

3.4 複数データフローに起因するネットワークトラフィックの問題点

複数データフローに起因するネットワークトラフィックは，データフロー数の増加により増大するという特徴を持つ．また，複数フローに起因するネットワークトラフィックの問題は，送信ノードから管理することが困難なことである．本項では以降，それぞれに関して説明する．

3.4.1 データフロー数の増加により発生するネットワークトラフィック

フラッディングされるデータフロー数の増加につれて増大するネットワークトラフィックに関して説明する．データフロー数とネットワークトラフィックとの関係は式 3.2に示したように正の相関関係を持つ．ここでは，フラッディングされる複数のデータフローがどのような環境において，ネットワークトラフィックを増大させるか図 3.5を用いて説明する．

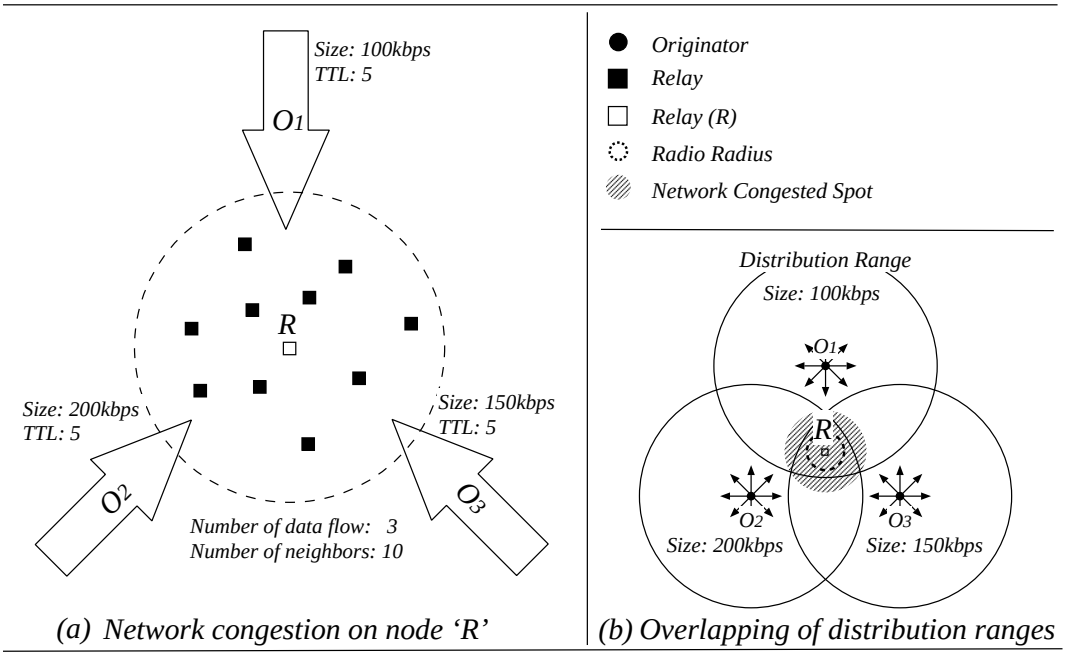

図 3.5: データフロー数の増加につれて増大するネットワークトラフィック

図 3.5(a)(b) は，複数のデータフローがフラッディングされることにより，ネットワークトラフィックがコンテンツ伝搬範囲の重複地点に集中することを示している．図 (b) には，データフローをフラッディングする送信ノード (O_1，O_2，O_3) の配置，伝搬範囲，さらに伝搬範囲の重複地点を示す．図 (a) には，伝搬範囲の重複地点に配置されるノード R の隣接ノード数，受信するデータフロー数を示す．また，送信ノード (O_1，O_2，O_3) の送信するデータフローがそれぞれ，100Kbps，200Kbps，150Kbps であること，図 3.5(a) のノード R の隣接ノード数が 10 であることを，式 3.2に代入することにより，ノード Re に対するネットワークトラフィックが 5Mbps 以上であることがわかる．

ここで，本研究の提案アプリケーションが街中などノードの密集したネットワークで利用されることを考慮すると，図 3.5に示すような無線範囲の重複数だけでなく，伝搬範囲の重複地点も増加することが予想される．そのため，送信ノードの密度が高い環境では，図 3.5以上に大量のネットワークトラフィックが発生し，無線の利用可能帯域を超える過剰なネットワークトラフィックが発生する可能性が高い．

3.4.2 ネットワークトラフィックの集中地点の予測困難性

複数データフローに起因するネットワークトラフィックの問題とは，複数のネットワークトラフィックの集中する地点が事前に予測出来ないことである．この問題により，データフロー数の増加により発生する過剰なネットワークトラフィックは事前に防ぐことが困難となる．本項では以降，複数データフローに起因するネットワークトラフィックの集中地点の予測困難性について説明する． 複数データフローに起因するネットワークトラフィックは，送信ノードの伝搬範囲が重複する地点に集中する．したがって，ネットワークトラフィックの集中する地点は，伝搬範囲の重複地点である．伝搬範囲の重複地点を予測するために必要な情報として，次の 2 つが考察される．

- 送信ノードの位置情報
- 送信ノードの伝搬範囲 (TTL)

送信ノードの位置情報と伝搬範囲を利用することにより，その送信ノードを中心とした伝搬範囲を予測することが出来る．ノードは各送信ノードの伝搬範囲を予測することで，どの地点にどのくらいの伝搬範囲が重複しているかを予測することが出来る．これにより，ネットワークトラフィックの発生地点を予測出来る．

しかし，TTL から予測される伝搬範囲が曖昧なことを考慮すると，正確なネットワークトラフィックの集中地点は予測出来ない．さらに，各送信ノードの位置情報と伝搬範囲が，集中地点を予測するノード全てに伝達される必要があるが，この処理により，各ノードの通信帯域が多く消費されてしまう．

以上示したように，ネットワークトラフィックの集中地点を予測するためには，複雑な処理が要求される．また，伝搬範囲から予測される範囲の曖昧さにより，正確な地点が予測出来ない．このように，ネットワークトラフィックの予測は困難である．

3.5 ネットワークトラフィックの抑制方針

本節では，フラッディングにより発生する過剰なネットワークトラフィックに対する本研究のアプローチを示す．3.2項において，フラッディング利用時に発生するネットワークトラフィック (T) が，隣接する中継ノード数 (N)，データフロー数 (n)，各データフローのデータサイズ ($x_k p_k$) と正の相関関係を持つことを示した．これらの値を減少させることにより，フラッディング利用時に発生するネットワークトラフィックを抑制することが出来る．本節では以降，これら 3 つに対して適用可能な処理を考察した後，本研究のアプローチを考察する．

3.5.1 ネットワークトラフィックの抑制処理に関する考察

表 3.1は N，n，$x_k p_k$ に対して適用可能な処理を示している．本項では N，n，$x_k p_k$ それぞれを減少させるために適用可能な処理に関して説明する．

表 3.1: 発生要因に注目したネットワークトラフィック (T) の抑制手法

Tを発生させる要因	Tの抑制手法
中継ノード数 (N)	中継ノードの選択
データフロー数 (n)	伝搬範囲の縮小，データフローの送信・再送制御
データフローサイズ ($x_k p_k$)	送信間隔の変更，パケットサンプリング，データ間引き

中継ノード数 (N) の削減処理

隣接する中継ノード数を削減する場合，不用意に削減が行われることによる伝搬範囲の縮小を考慮する必要がある．そのため，隣接する中継ノード数を削減するためには，3.3節で説明した，必須ノード，冗長ノードに注目して，必須ノードを中継ノードとして選択する必要がある．また，この処理は中継ノードを予め選択しておく手法であり，ネットワークトラフィックの発生に関わらず行われる必要がある．

データフロー数 (n) の削減処理

データフロー数を削減する場合はデータフローの送信制御，伝搬範囲の縮小，データフローの再送制御，3 つの処理が挙げられる．前者 2 つは送信ノード，後者は中継ノードによる処理である．以降それぞれに関して説明する．

まず，データフローの送信制御処理に関して考察する．この処理は送信ノードが伝搬範囲内における過剰なネットワークトラフィックを予測・検知した上で行われるべきである．この処理は複数の送信ノードが同時にネットワークトラフィックを予測・検知した場合，一斉にデータフローの送信が終了するという問題を持つ．

次に，データフローの伝搬範囲 (TTL) を狭める処理に関して考察する．この処理は，過剰なネットワークトラフィックの発生している地点が伝搬範囲に含まれないように，伝搬範囲を狭めることで，その地点におけるデータフロー数を削減する．この処理もネットワークトラフィックを予測・検知した上で行われるべきである．また，データフローの送信を停止する方法と同様に，一斉にデータフローの伝搬範囲が狭まるという問題がある．

最後にデータフローの再送制御処理に関して考察する．この処理は，中継ノードにおいて，過剰なネットワークトラフィックが予測または検知された場合に行われるべきである．この処理の特徴は，中継ノードがデータフローの再送を制御することで，伝搬範囲の一部に対してのみデータフローが伝搬されないことである．これは，送信ノードによる処理が伝搬範囲全体に影響することと比較して有利な点である．

データフローサイズ ($x_k p_k$) の削減処理

データフローサイズを削減に関しては，送信間隔の変更やパケットサンプリング，データ間引きなどの処理が適用出来る．これらの処理は送信ノード及び中継ノードで利用可能である．

まず，パケット送信間隔の変更に関して考察する．送信間隔の変更は利用可能なアプリケーションが限定されており，例えば一定の帯域を常時利用するリアルタイムストリーミングなどのコンテンツ配信には適用出来ない．この処理は，データサイズがあらかじめ定められている

コンテンツの配信において利用されるべきである．

次に，パケットサンプリング及びデータ間引きに関して考察する．これらの処理はコンテンツを構成するデータ量を削減するため，受信するコンテンツのデータ品質や内容が元のコンテンツよりも劣化，減少する．そのため，不用意に利用されるべきではなく，送信ノードや中継ノードが過剰なネットワークトラフィックを予測・検知した場合にのみ利用されるべきである．

3.5.2 ネットワークトラフィックの抑制に対する本研究のアプローチ

以上，フラッディング利用時に発生するネットワークトラフィックを抑制可能な処理に関して考察した．これらの処理は，行動主体別に表3.2のように分類できる．次に，ネットワークトラフィックの抑制に対して本研究のとるアプローチを行動主体に注目し考察する．

表 3.2: 取りうるアプローチの行動主体別分類

行動主体	アプローチ
全ノード	隣接する中継ノード数の選択処理
送信ノード	伝搬範囲の縮小，データフローの送信制御 送信間隔の変更，パケットサンプリング，データ間引き
中継ノード	送信間隔の変更，データフローの再送制御 パケットサンプリング，データ間引き

以上分類した処理のうち，全ノードのアプローチに関しては，ネットワークトラフィックの予測または検知に関わらず行われるべき処理である．一方で，送信ノード及び中継ノードによるアプローチは，ネットワークトラフィックの予測または検知が行われた上で利用されるべき処理である．さらに，送信ノードと受信ノードの行う処理では，影響の及ぶ範囲が伝搬範囲全体か，一部かという違いがある．次に，ネットワークトラフィックの予測・検知，処理の影響する範囲に注目して送信ノード，受信ノードの処理を比較する．

まず，ネットワークトラフィックの予測・検知に注目する．予測に関して考察すると，3.4.2項にて示したように，送信ノード，受信ノード共に，過剰なネットワークトラフィックの予測は困難である．次に検知に関して考察すると，送信ノードの処理に関しては伝搬範囲全体に影響するため，伝搬範囲内における過剰なネットワークトラフィック全てを検知する必要がある．しかし，送信ノード自体が検知可能なネットワークトラフィックは，送信ノードの配置される地点に限られる．そのため，過剰なネットワークトラフィックの発生する地点に配置されるノードから，過剰なネットワークトラフィックを検知したことが通知されるなどの処理が必要になる．一方で，中継ノードでは検知するべきネットワークトラフィックは中継ノードの配置される地点におけるネットワークトラフィックであるため，容易に過剰なネットワークトラフィックを検知出来る．

次に，処理の影響する伝搬範囲の違いに注目する．送信ノードでは伝搬範囲全体に影響があるため，過剰なネットワークトラフィックが発生していない地点に対しても，データフローの送信制御，データ品質の劣化を要求する．したがって，各地点に適した処理が出来ない．一方で，中継ノードでは中継ノード以降の伝搬範囲にのみ影響を及ぼすため，過剰なネットワークトラフィックの発生している地点に関連する範囲のみにデータフローの再送制御，データ品質

の劣化などを要求する．このように，送信ノードと比較して中継ノードの方がより，環境に応じたネットワークトラフィックの抑制が可能である．

以上，送信ノードと中継ノードを，ネットワークトラフィックの予測・検知，処理の影響する範囲に関して比較を行った．この結果，中継ノードでネットワークトラフィックを抑制する手法の方が，実現性，規模性ともに高いことがわかる．そのため，本研究では，中継ノードにおけるネットワークトラフィックの抑制処理を利用する．なお，中継ノードにおける抑制処理として 4 つの処理を示したが，このうち送信間隔の変更処理に関してはアプローチの対象外とする．これは第 2章にて提案したサービスモデルではテキストや画像といったデータサイズが固定されたコンテンツ以外に，リアルタイムストリーミングのような常時一定の帯域を消費するコンテンツが想定されるためである．

3.6 本章のまとめ

本章では，本研究の提案モデルを実現するために考慮すべき問題点として，フラッディングの利用によるネットワークトラフィックの増大に注目した．本章では，このネットワークトラフィックを抑制するために，フラッディングの特徴及び定義を示した上で，ネットワークトラフィックを抑制するために必要なアプローチに関して議論した．これらの結果を基に，ネットワークトラフィックの抑制方針として，全ノードによる中継ノードの選択や中継ノードによるデータフロー制御を利用することを示した．次章では，本章で示した抑制方針に沿って，既存技術の紹介を行う．

第4章　ネットワークトラフィックの抑制方針に関連する既存技術

第3章にて，全ノードの利用すべきアプローチとして中継ノードの選択処理，中継ノードの利用すべきアプローチとしてデータフローの再送制御，パケットサンプリング処理，データ間引き処理を示した．表4.1には，これらの抑制方針それぞれに関連する既存技術を示す．本章では以降，表4.1に示す既存技術それぞれについて紹介し，抑制方針との関連性を説明する．

表 4.1: 本研究のアプローチに対応する既存技術

対応する指標	行動主体	アプローチ	関連する既存技術
N	全ノード	中継ノードの選択	Multipoint Relay
n	中継ノード	データフローの再送制御	Differentiated Service
$x_k p_k$	中継ノード	パケットのサンプリング処理	Layered Multicast
	中継ノード	データ間引き処理	DV Stream Control Proxy

4.1　中継ノードの選択に関連する既存技術

中継ノードの選択に関連する既存技術としてMultipoint Relay(MPR)[8] [16][17]やEssencial Connecting Dominating Set(E-CDS)，MPR-CDS[18]など様々なアルゴリズムがこれまでに提案されている．これらのアルゴリズムはアドホックネットワークでの利用を想定して提案されており，特にルーティングプロトコルにおける経路情報交換の効率化に利用される．また，これらのアルゴリズムを利用した配信技術としてSimplified Multicast Forwarding[9]が提案されている．本節では以降，中継ノード選択のアルゴリズムとしてMPR，配信手法としてSMFを説明する．

4.1.1　Multipoint Relay Set Selection Algorithm

これまでに，中継ノードの選択アルゴリズムとしてトポロジ情報やクラスタ情報，再送確率，位置情報等をベースにしたアルゴリズムが提案されている．これらの中で，MPRはトポロジ情報を利用するアルゴリズムである．MPRに注目する理由は，アルゴリズムの効率性が多くの研究により示されていること，再送確率や位置情報を利用するアルゴリズムに比べ確実に伝搬範囲を拡大出来ることの2点である．MPRは，各ノードに対する2ホップいないのトポロジ情報を利用することにより，中継ノードの選択を行う．中継ノードの選択は以下の手順により行われる．

1. HELLOメッセージを利用したトポロジ情報の構築

2. トポロジ情報を基にした必須ノードの選択

まず，HELLO メッセージを利用したトポロジ情報の構築に関して説明する．MPR を利用するノードは HELLO メッセージを定期的にブロードキャストする．初めの段階では，ノード自身のアドレスが格納された HELLO メッセージが送信される．この HELLO メッセージを受信することにより，隣接ノードは HELLO メッセージを送信するノードのアドレスを取得する．次の段階では，ノード自身のアドレスに加え，隣接ノードのアドレスが格納された HELLO メッセージが送信される．この HELLO メッセージを受信することにより，隣接ノードは HELLO メッセージを送信するノードのアドレスに加え，そのノードの隣接ノードのアドレスを取得する．このように，各ノードは HELLO メッセージの交換により，2 ホップ以内に配置されるノードのアドレス，2 ホップ先のノードと隣接ノードとのリンク関係を把握することが出来る．

次に，構築されたトポロジ情報を基にした必須ノードの選択に関して説明する．説明では，隣接ノードを 1 ホップノード，2 ホップ先のノードを 2 ホップノードとする．初めの段階では，2 ホップノードとリンク関係にある 1 ホップノードが 1 台のみである場合，その 1 ホップノードを必須ノードとして選択する．次の段階では，選択した必須ノードとリンク状態にある 2 ホップノードに注目し，これらの 2 ホップノード以外とリンク関係を持たない 1 ホップノードを冗長ノードとして選択する．最後に，既に選択された必須ノードと冗長ノード以外の 1 ホップノードの中から，リンク関係にある 2 ホップノード数が最大のノードを必須ノードとして選択する．この処理は選択されない 2 ホップノードがなくなるまで行われる．

以上，HELLO メッセージを利用したトポロジ情報の構築，トポロジ情報を基にした必須ノードの選択を説明した．これらの処理の後，各ノードは選択された必須ノードのアドレスを HELLO メッセージに格納し送信する．これにより，隣接ノードは自身が必須ノードであるかを検知する．これら一連の処理により実現されるフラッディングと，単純なフラッディングの違いを図 4.1に示す.

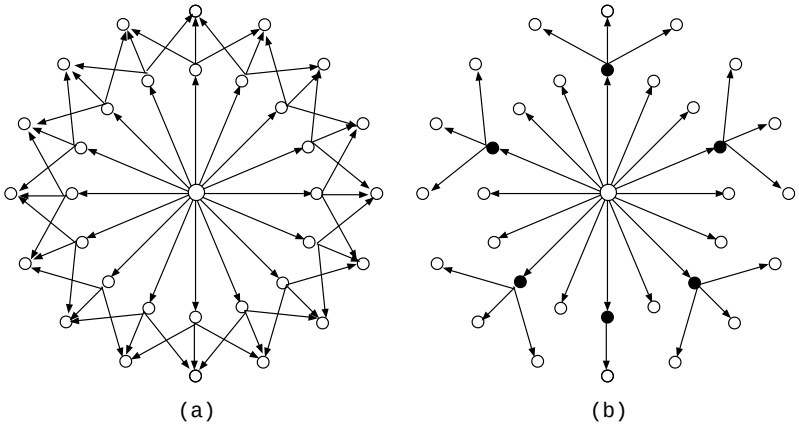

図 4.1: Classical Flooding(a) vs. MPR Flooding(b)

図 (a) は単純なフラッディング，図 (b) は MPR を利用したフラッディングを示す．また，図 4.1の矢印は，パケットの伝搬の流れを示している．これら 2 つの図を比較すると，図 (a) では 16 台，図 (b) では 6 台が中継ノードとして選択されている．これは，3.1式より，図 (a) で発生するネットワークトラフィックが，図 (b) の 2〜3 倍であることを示している．このように，MPR を利用することにより，フラッディング利用時に発生するネットワークトラフィックを効率的

に抑制することが出来る．

4.1.2 Simplified Multicast Forwarding

SMF は，中継ノードの選択アルゴリズムを利用して効率的なフラッディングを実現すること，マルチキャストにおけるコントロールパケットを効率的に伝搬することの 2 点を目的に提案された配信技術である．SMF は，マルチキャスト技術への適用が考慮されているものの，パケットの種類に依存せず効率適なフラッディングを実現する汎用的なフレームワークである．SMF は IETF の MANET ワーキンググループで標準化が進められている技術であり，仕様やプロトコルなど詳細に関して議論が行われている．本節では，現段階の仕様において定義されている機能を説明する．SMF に定義されている機能は，中継ノードの選択と重複パケットの検知の 2 つである．

まず，中継ノードの選択に関して説明する．SMF では，中継ノードの選択アルゴリズムとして MPR や E-CDS，MPR-CDS などの利用が考慮されている．前項にて，MPR が HELLO メッセージを利用することを示したが，SMF では，MPR だけでなく様々な中継ノードアルゴリズムに適応するため，HELLO メッセージ以外に Neighbor Discovery Protocol(NDP)[19] を利用して中継ノードの選択処理を行う．SMF における，中継ノードの選択に関連する機能の論理構成図を以下に示す．図 4.2において，*付きのメッセージは，中継ノード選択アルゴリズムに利用される可能性のある情報を示している．

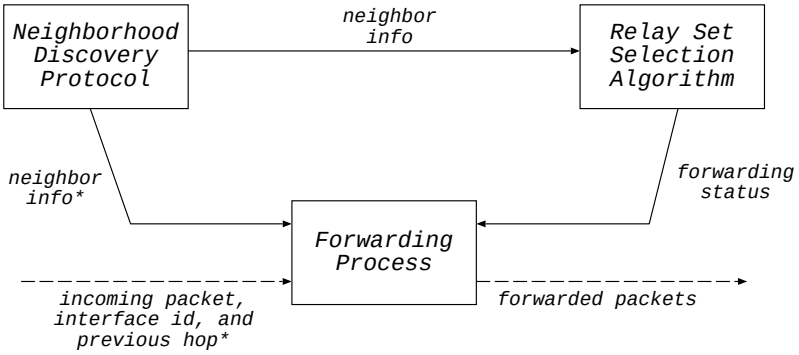

図 4.2: SMF Node Architecture

次に，重複パケットの検知に関して説明する．通常，重複パケットの検知は送信先・送信元アドレスやパケットの識別子などを利用して行われる．アドホックネットワークにおけるルーティングプロトコルが持つ重複パケットの検知機能では，パケットの識別子がデータパケットに記述される．そのため，重複パケットの検知は，プロトコルに依存する．SMF はルーティングプロトコルやマルチキャストなど，様々なプロトコルに対応するために，重複パケットの検知を，IP ヘッダの拡張ヘッダを利用して行う．IPv4 の場合は ID フィールド [20] に，IPv6 の場合は Hop-by-Hop オプションヘッダ [21] にパケットの識別子が記述される．SMF は IP ヘッダを利用することにより，プロトコルに依存することなく，重複パケットの検知を実現する．
以上，中継ノードの選択に関連する技術として MPR 及び SMF を紹介した．両技術ともにアドホックネットワークでの利用が想定される技術であり，中継ノード数の削減を実現することでネットワークトラフィックの抑制を実現する技術といえる．

4.2 データフローの再送制御に関連する既存技術

データフローの再送制御に関連技術として，Differentiated Service(DiffServ) [22] が挙げられる．DiffServ は送受信するデータフローの種類や優先度などを識別することにより，複数の優先クラス間で相対的な転送性能差をつける技術である．このように，あらかじめ転送性能差をつけることにより，過剰なネットワークトラフィックが発生した場合に，優先度の高いデータフローの転送処理を優先的に行うことが出来る．これらの処理を行うことにより，DiffServ は過剰なネットワークトラフィックが発生した場合にも，優先度やデータ種類に応じたサービス品質 (QoS) を保証出来る．本節では以降，DiffServ を利用することで，データフローがどのように優先順位付けされるか，説明する．

DiffServ は Differentiated Service Codepoint(DSCP)[23] と呼ばれる 6bit のフィールドを利用することにより，データフローの優先順位付けを行う．DSCP フィールドは，IPv4 では Type of Service(TOS) フィールド，IPv6 では Traffic Class フィールドに割り当てられる．DiffServ は DSCP フィールドを定義するヘッダとして，Differentiated Service(DS) ヘッダを定義している．DS ヘッダの定義を図 4.3に示す．

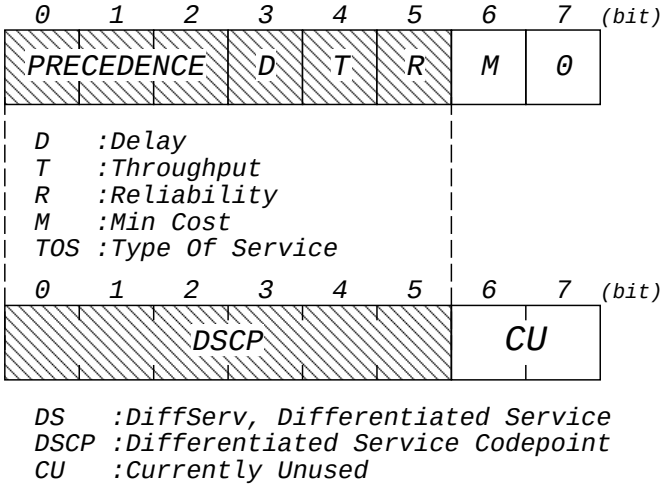

図 4.3: Differentiated Service ヘッダ

図 4.3に示すように，DS ヘッダの上位 6bit に DSCP フィールドが定義されている．また，下位 2bit には，Early Congestion Notification(ECN:初期輻輳通知)[24][25] フィールドが予約されている．DSCP の上位 3bit は優先度，続く 3bit はそれぞれ遅延，通信速度，信頼性を示す．DiffServ はこれら 4 つフィールドを参照することにより，2 段階の優先順位付けを行う．

初めの段階では，DSCP フィールドのうち，優先度が利用される．DiffServ は，パケット毎に優先度を参照することにより，データフローを優先度毎にクラス分けする．また優先度以外に，ポート番号や受信インタフェース，パケット長など，DS ヘッダ以外の情報が参照される場合がある．次の段階では，DSCP フィールドのうち，下位 3bit が利用される．DiffServ は，下位 3bit を参照することにより，同じクラスのデータフローを差別化および優先順位付けを行う．このようにして，優先度に応じたクラス分け，クラス内における遅延や通信速度，信頼性に応じた優先順位付け，2 段階の優先順位付けが行われる．

以上，複数データフローの再送制御に関連する技術として DiffServ を紹介した．これまでに，

DiffServ を利用することでアドホックネットワークにおける QoS の実現を目的とした研究 [26] が行われている．これらの研究により，アドホックネットワークにおいても DiffServ を利用することにより優先度の高いデータフローを優先的に再送することが可能であることが示されている．このように複数データフローの再送制御を実現するためには，優先度等を利用したデータフロー管理が有効である．DiffServ の目的は QoS を実現することであり，ネットワークトラフィックの抑制ではないが，優先度を利用したデータフロー管理手法は本研究の参考にすべき点がある．

4.3 パケットサンプリング処理に関連する既存技術

パケットサンプリング処理では，送信されるデータフローの各パケットが映像・音声フレームや，付録 Aに示す階層符号化されたデータのレイヤ群など，意味のあるデータ単位で送信されることが必要である．さらに，各パケットのグループが明示される必要がある．これらの条件の下，パケットをグループ単位で棄却することにより，再生可能なデータのまま，データフローサイズの削減が実現出来る．本節では以降，パケットサンプリング処理に関連する既存技術として Layered Multicast(LM)[14][15] を紹介する．

LM は階層符号化されたデータを利用することで，受信者のネットワーク環境に適応したマルチキャスト配信を実現する．LM では階層符号化されたデータの Base レイヤや Enhanced レイヤが，異なるマルチキャストアドレス宛に送信される．このように，各パケットをレイヤに応じてグルーピングして送信することにより，参加するマルチキャストグループ数に応じて，データ品質やデータフローサイズをコントロール出来る．このことを図 4.4を利用して説明する．

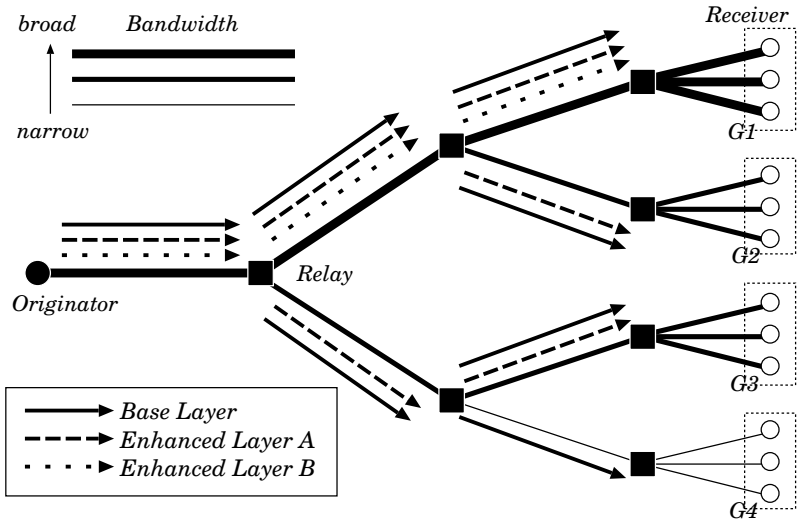

図 4.4: Layered Multicast によるパケットサンプリング処理

図 4.4は，階層符号化されたデータの Base レイヤ (BL)，Enhanced レイヤ (ELA，ELB) がそれぞれ，異なるマルチキャストアドレス宛に送信される環境を示す．また，図 4.4の G1-G4 はそれぞれ異なるネットワークに属する受信ノード群を示す．図 4.4では G1-G4 により，それぞれのネットワーク環境に応じたマルチキャストグループ数が選択されている．ここで，各レイ

ヤ毎にパケットがグルーピングされていることを考慮すると，参加するマルチキャストグループ数を減少することにより，受信するデータフローのデータフローサイズを削減することが出来る．

以上，パケットサンプリング処理に関連する既存技術として LM を紹介した．この処理を利用するためには，各パケットを映像フレームといったデータ単位でグルーピングして送信すること，さらにそれらのパケットからデータを再構成できることがアプリケーションに要求される．パケットサンプリング処理は，アプリケーションに対する制約を持つが，一方で制約条件を満たしたアプリケーションのデータフローサイズをアプリケーション非依存に削減出来るという利点を持つ．本研究ではこの利点に注目し，ネットワークトラフィックの抑制手法として，パケットサンプリング処理を利用する．

4.4 データ間引き処理に関連する既存技術

各データフローのデータ間引き処理は，データフローを構成する各パケットのデータを一部棄却することにより行われる．この処理は映像・音声フレームや階層符号化技術を利用したデータのレイヤ群など，意味のあるデータ単位で行われる．したがって，各パケットのデータを解析すること，どのデータ部分が間引き対象となるデータ単位に含まれるか判別することの 2 点が必要となる．これらを実現するためにはデータ構造に対応した解析処理が必要となる．そのため，パケットサイズの削減処理にはアプリケーションに依存した処理が必要となる．本節では以降，パケットサイズの削減を実現する関連技術として，DV Stream Control Proxy(DSCP)[27] を紹介する．

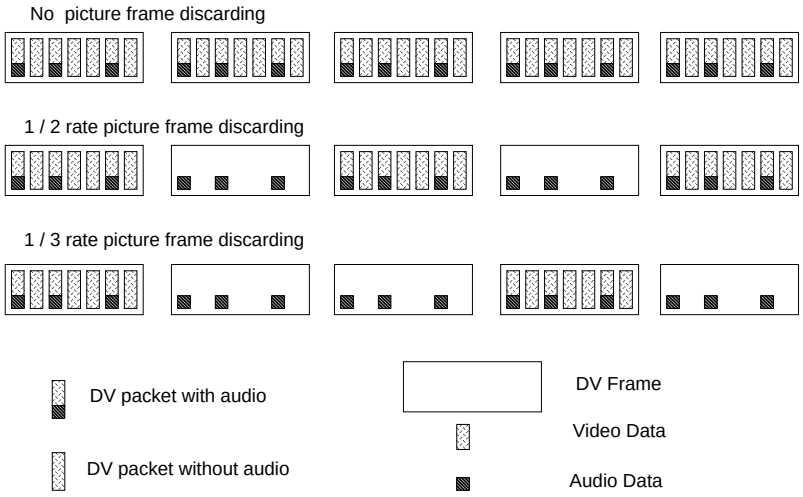

図 4.5: DV データのフレーム間引き処理

DSCP は映像・音声転送システムである Digital Video Transport System(DVTS)[28] の中継ノードとして機能し，受信した DV データの間引き処理を実現する技術である．DV データは図 4.5に示すように，映像フレームと音声フレームから構成される．DSCP は DV データを構成するパケットを受信すると，各パケットをアプリケーションレイヤに上げ，データの間引き処理を行う．この際，あらかじめ設定したフレームレートに応じて間引き対象のフレームを決

定する．DSCP によるデータ間引き処理を図 4.5に示す．

図 4.5は上から順に，DSCP に設定されるフレームレートがフルレート，ハーフレート，1/3 レート，それぞれの場合においてどのように DV データが間引かれるかを示す．図 4.5中の長方形の枠は映像フレームを，黒い正方形の枠は音声フレームを示す．図 4.5には，音声フレームのみの DV フレームが描かれているが，これは DSCP による映像フレームのデータ間引き処理が行われたことを示す．また，DSCP は一定以上のデータ量が送信バッファに溜ることで DV パケットを送信する．

以上，データ間引きに関連する既存技術として DSCP を紹介した．既に示したようにデータ間引き処理はアプリケーションに依存する処理である．従って，データ間引き処理に関しては汎用的な処理方法を提案することが困難である．また，データ間引き処理は，パケット毎にデータ解析が必要であるため計算機資源やアプリケーションバッファを多く消費されるという特徴を持つ．一方でデータ構造に適した処理が可能であるため，パケットサンプリング処理よりも，詳細なデータフロー削減が可能であるという利点を持つ．本研究ではこの利点に注目し，ネットワークトラフィックの抑制手法としてデータ間引き処理を利用する．

4.5 本章のまとめ

本章では，ネットワークトラフィックの抑制方針である，中継ノードの選択処理，データフロー数の削減処理，パケットサンプリング処理，データ間引き処理それぞれに関連する既存技術を紹介した．次章では，これらの技術を応用することでネットワークトラフィックの抑制を実現するコンテンツフラッディングモデルを提案する．

第5章　Adaptive Application Flooding

第 2章にて提案したサービスモデルを実現するためにはフラッディング利用時に発生するネットワークトラフィックの抑制が必要である．本章では，第 3，4章にて示した抑制方針に基づきネットワークトラフィックを抑制するコンテンツフラッディングモデルとして，Adaptive Application Flooding を提案する．

5.1　Adaptive Application Flooding の概要

第 2章で提案した広告配信モデルを実現するためには，フラッディング利用時に発生するネットワークトラフィックの抑制が不可欠である．第 3，4章では，ネットワークトラフィックの抑制方針として中継ノードの選択，データフロー数の削減，パケットサンプリング処理，データ間引き処理の必要性を述べ，それぞれに参考となる既存技術を紹介した．

本研究では，これらの抑制処理に注目したコンテンツフラッディングモデルとして Adaptive Application Flooding(AAF) を提案する．AAF は配信機構の役割を持つ Delivery module(DM)，フローコントロール機構の役割を持つ Autonomous Flow Management module(AFM) から構成される．まず，AAF を利用することによりフラッディング利用時に発生するネットワークトラフィックの抑制がどのように実現されるか図 5.1を用いて説明する．

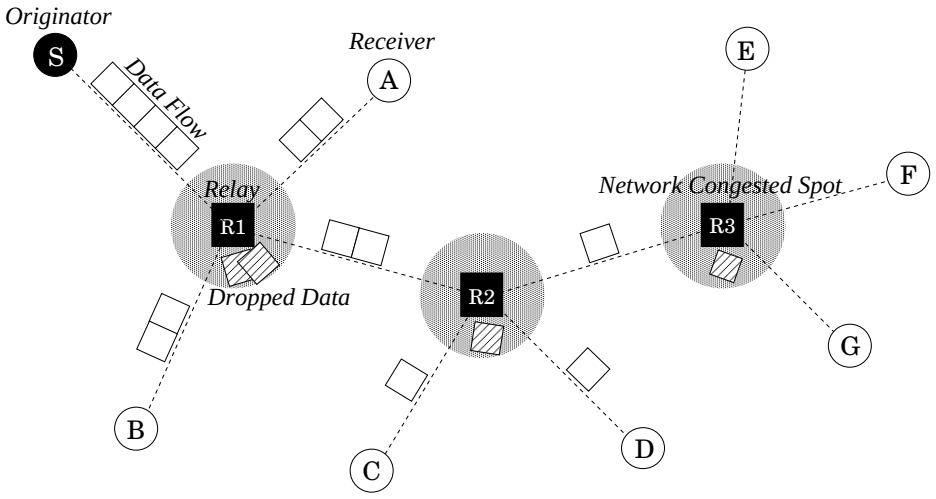

図 5.1: Adaptive Application Flooding の動作概要

図 5.1は送信ノード S からフラッディングされる，データフローサイズの削減可能なデータが，ネットワーク状態に応じて削減されていく様子を示している．図 5.1に示す白のブロックは送信ノード S によりフラッディングされるデータを示す．また図 5.1では，中継ノード R1-R3

において過剰なネットワークトラフィックが発生している．ここで中継ノード R1-R3 の再送するデータに注目すると，中継ノードがデータを再送するにつれてデータフローサイズが削減されていくことがわかる．このように，AAF を利用してフラッディングされるデータはネットワーク状態に応じてデータフローサイズが削減される，もしくは棄却されるという特徴を持つ．このように，AAF はネットワーク状態に合わせてデータフローサイズを削減することにより，過剰なネットワークトラフィックを抑制する．次に AAF を構成する DM と AFM の動作概要及び相互関係を図 5.2を利用して説明する．

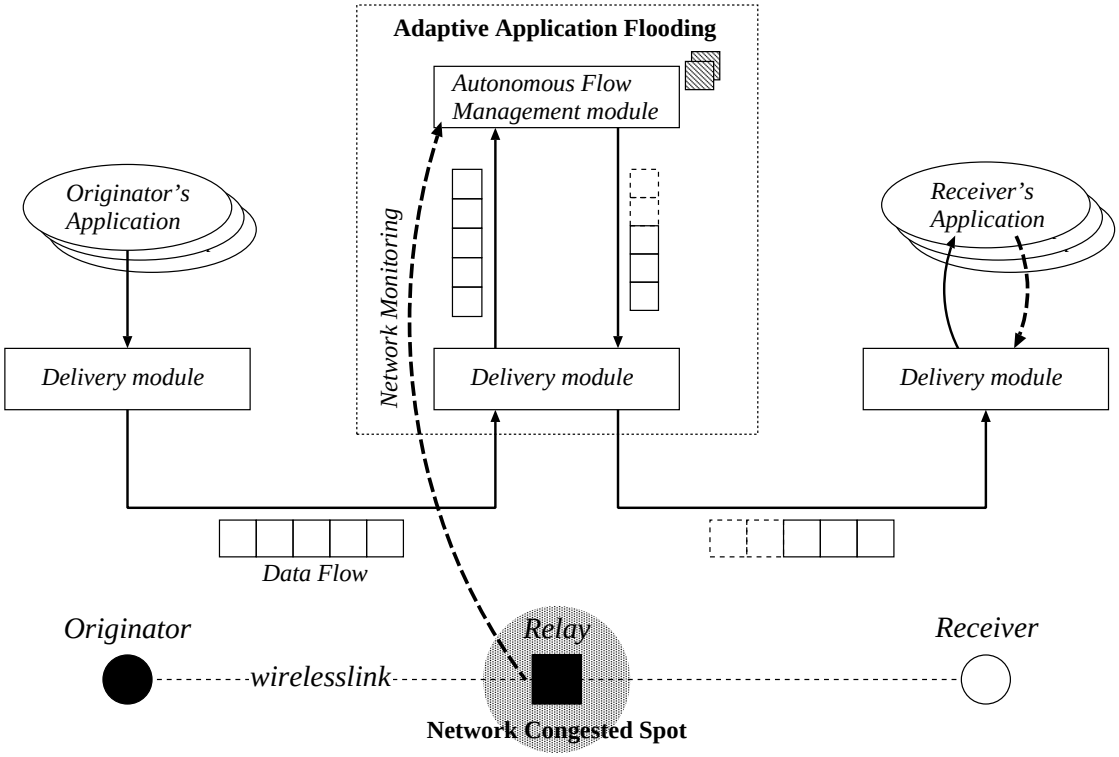

図 5.2: Adaptive Application Flooding

- Delivery module(DM)
  DM は中継ノードの選択によりフラッディング利用時に発生するネットワークトラフィックを抑制する機能を持つ．DM は全ノードにより利用されるモジュールであり，AAF において配信機構の役割を持つ．DM は隣接ノードで動作する AAF の DM とメッセージを交換することにより，中継ノードの選択処理を実現する．この機能により，DM は冗長ノードによるパケットの再送を防ぎ，フラッディング利用時に発生するネットワークトラフィックを抑制する．DM は中継ノードの選択以外に，複数のユーザアプリケーションから利用される機能を持つ．また，DM はアプリケーションデータを構成するパケットを受信した場合，そのパケットのデータ部分を AFM に渡し，AFM からデータが返された場合はパケットヘッダを付随してネットワークに転送する．

- Autonomous Flow Management module(AFM)
  AFM はデータフローの再送制御やデータフローサイズの削減により過剰なネットワーク

トラフィックの発生を回避する機能を持つ．AFM は中継ノードにより利用されるモジュールであり，AAF におけるフローコントロール機構として動作する．AFM は DM から渡されたデータをアプリケーションレイヤにて処理する．また，AFM はネットワークトラフィックを常時監視することにより，ネットワーク状態の変化を検知する．AFM はネットワーク状態の変化を検知すると，データに設定された優先度に応じて処理対象を決定する．AFM は処理対象として選択されたデータフローに対して，再送制御やデータフローサイズ削減処理を適用する．AFM はデータを処理した後，処理されたデータを DM へ返す．

以上，AAF を構成する DM と AFM の動作概要及び相互関係を示した．AAF は，これらのモジュールの連係よりネットワークトラフィックを考慮したコンテンツフラッディングを実現する．本章では以降，DM 及び AFM の提供する各機能について説明する．

5.2 Delivery module の提供する機能

DM の提供する機能は次の 5 つである．

1. 中継ノードの選択処理
2. 隣接ノードの Delivery module とのパケット交換機能
3. Autonomous Flow Management module とのデータ交換機能
4. アプリケーションからネットワークへのデータ中継機能
5. ネットワークからアプリケーションへのデータ転送機能

これら 4 つの機能のうち，(1)-(3) の機能はネットワークトラフィックの抑制を目的とする機能に分類される．このうち (1)，(2) の機能は効率的なフラッディング手法を利用するための条件である．本研究では (1)，(2) の機能に関して既存手法である SMF を利用する．さらに SMF に (3) の機能追加することにより，AAF における再送制御やデータフローサイズ削減が可能となる．一方で (4)，(5) の機能は複数アプリケーションからの利用を目的とする機能である．これらの機能は，第 2章で提案したサービスモデルが種類の異なる複数のアプリケーションから利用されるために必要となる．DM はこれらの機能を実現するために，各パケットのヘッダに DM ヘッダを付随することを要求する．

5.3 Autonomous Flow Management module の提供する機能

AFM の提供する機能は次の 4 つである．

1. データフローの優先度を利用したグループ管理機能
2. ネットワークトラフィックの監視によるネットワーク状態検知機能
3. ネットワーク状態及び優先度に応じたデータフローの再送制御機能
4. ネットワーク状態及び優先度に応じたデータフローサイズ削減機能

これらの機能のうち，(1)，(2) の機能は過剰なネットワークトラフィックの発生を検知するために，(3)，(4) は検知を基にしたデータフロー数，データフローサイズの削減を行うために利用される．本節では以降，AFM の扱う優先度の特徴を説明する．その後，AFM の再送制御，データフローサイズ削減がどのように行われるか説明する．

5.3.1 Autonomous Flow Management における優先度の特徴

AFM を利用するためには，AFM を利用するデータフローの各パケット内に明示的に優先度を記述する必要がある．AFM は優先度を記述するためのデータヘッダとして AFM ヘッダを持つ．AFM は DM からデータが渡される毎に AFM ヘッダを参照することにより，各データフローを優先度に対応した優先グループに分類する．AFM は過剰なネットワークトラフィックの検知時に，優先度の低い優先グループから再送制御，データフローサイズ削減の対象とする．

AFM における優先度の特徴は，優先度が中継ノードを経由するにつれて次第に低くなることである．この特徴により，過剰なネットワークトラフィックが発生する環境において，より多くの中継ノードを経由したデータフローから，再送制御やデータフローサイズ削減処理の対象に選択される．したがって，送信された AAF を利用して送信されるデータフローは，送信ノード付近において優先的に再送される．

5.3.2 ネットワーク状態及び優先度に応じたデータフローの再送制御

AFM は，ネットワーク状態の変化に応じて処理対象とする優先グループを決定する．再送制御機能は処理対象とするデータフローのデータサイズ削減が不可能な場合に適用される．AFM による再送制御の様子を図 5.3に示す．

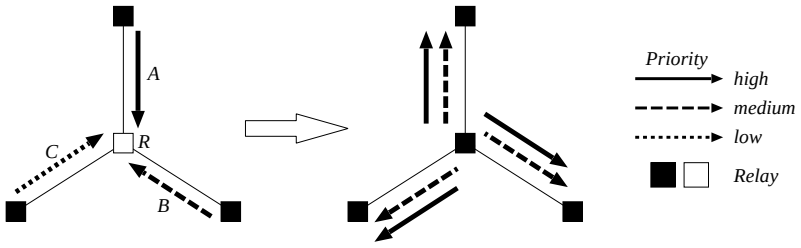

図 5.3: Adaptive Application Flooding によるデータフローの優先制御

図 5.3は，データフロー A，B，C がフラッディングされることにより，中継ノード R に対して，過剰なネットワークトラフィックが発生している状態を示す．図 5.3における白と黒のブロックはそれぞれ，過剰のネットワークトラフィックを検知している状態か否かを表す．図 5.3に示す環境において，AFM は優先度の低いデータフローを再送制御の対象とする．これにより，優先度の高いデータフローは優先的に再送され，低いデータフローは棄却されることになる．優先度に応じてデータフローの再送制御が行われることにより，図 5.3のように，R により再送されるデータフロー数が削減され，ネットワークトラフィックの抑制が実現される．

5.3.3 ネットワーク状態及び優先度に応じたデータフローサイズの削減機能

優先制御の処理対象となる優先グループ内に，データサイズの変更可能なデータフローが含まれていた場合，データフローサイズの削減処理が適用される．AFM はデータフローサイズの削減処理としてパケットサンプリング処理，データ間引き処理を利用する．以降ではそれぞれの処理に関して説明する．

パケットサンプリング処理

次に，パケットサンプリング処理に関して説明する．各データフローが優先度に応じて優先グループに分類されることを説明したが，パケットサンプリング処理はこの優先グループを利用して実現出来る．このことを図 5.4を利用して説明する．

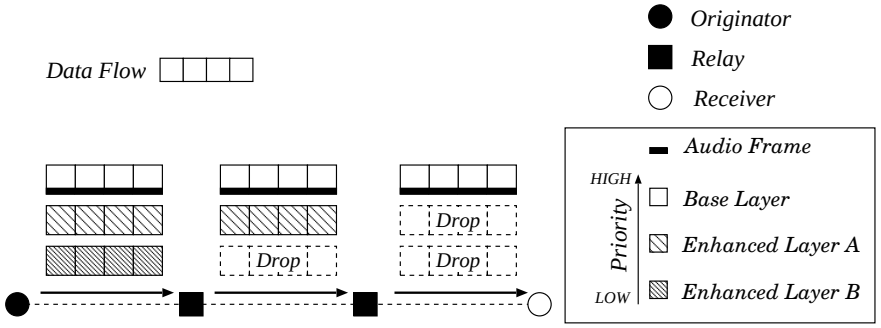

図 5.4: Autonomous Flow Management module によるパケットサンプリング処理

図 5.4は，階層符号化技術を利用した映像・音声データに対して AFM がパケットサンプリング処理を適用する様子を示す．図 5.4において注目すべき点は，階層符号化された映像フレームの Base レイヤ，Enhanced レイヤが優先度に差を付けて送信されていることである．これは一つのデータフローを優先度の異なる複数のデータフローに分割して送信することを意味する．既に述べたように，AFM は優先度の低いデータフローから再送制御処理を適用する．したがって過剰なネットワークトラフィックが検知されると，最も優先度の低いデータフローから棄却されていく．図 5.4ではまず，Enhanced レイヤ B を構成するデータフローが棄却されていることが分かる．このように，コンテンツを優先度の異なる複数のデータフローに分割して送信することにより，データフローサイズの削減が実現出来る．図 5.4では階層符号化技術を利用したデータを例に挙げたが，パケットサンプリング処理はテキストデータやフレーム内圧縮されたデータに関しても適用可能である．

データ間引き処理

次に，データ間引き処理に関して説明する．この処理は，各パケットのデータ間引きを行う処理である．データ間引きを行うためにはまず，コンテンツのデータ構造を把握する必要がある．AFM はデータ構造を把握するために AFM ヘッダにデータ構造を記述するためのフィールドを持つ．また，データ構造を把握した上で，それに対応したデータ解析や間引き処理が必要となるが，AFM はこれらの処理を追加するための拡張モジュールを持つ．拡張モジュールすることで実現されるデータ間引き処理を図 5.5に示す．

図 5.5では，データ間引き可能なデータ構造として，階層符号化技術が利用されている．過剰なネットワークトラフィックの検知時にこのデータフローが処理対象となった場合，AFM はデータを受け取る毎にそのデータ構造を解析し，棄却すべきデータセグメントを決定する．図 5.5では Enhanced レイヤ B を構成するデータから棄却されていることが分かる．以上説明したように，AFM は AFM ヘッダに記述されるデータ構造に応じたデータ解析を行うことで棄却対象となるデータセグメントを決定する．これによりデータ間引き処理が行われる．

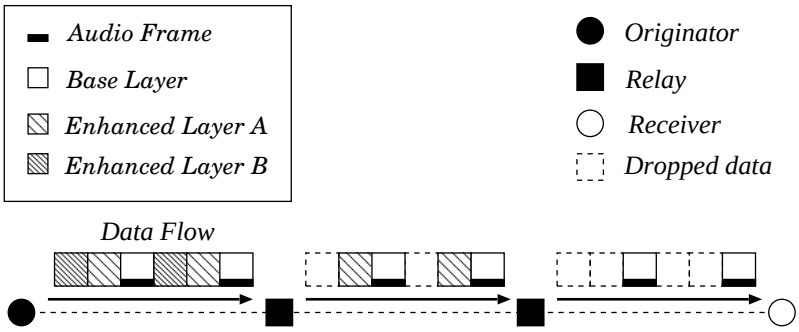

図 5.5: Autonomous Flow Management によるデータ間引き処理

5.4 Adaptive Application Floodingの対象とするアプリケーション

本節では，AAF が対象とするアプリケーションの特徴を示した後，AAF を利用するアプリケーションへの要求事項を説明する． AAF が対象とするアプリケーションに関して説明する．AAF は中継ノードの選択だけでなく，データフローの再送制御やデータフローサイズの削減処理によりフラッディング利用時に発生する過剰なネットワークトラフィックを回避する．そのため，AAF を利用する場合には，エンドノードへのデータ到達性や送受信するデータの同一性は保証されない．したがって，データ到達性や送受信するデータの同一性を前提とするフラッディングアプリケーションは AAF を利用するべきでない．一方で，AAF はコンテンツを構成する情報の中でも，送信者が最重要とする情報に関しては，出来る限り伝搬範囲全体に伝達するように動作する．このことから AAF はコンテンツの中でも最低限必要となる情報を，可能な限り多くのノードに伝搬することを重要とするフラッディングアプリケーションから利用されるべきである．

AAF を利用するアプリケーションへの要求事項を説明する．AAF は DM，AFM から構成される．既に説明したように，これらのモジュールはそれぞれ，データヘッダとして DM ヘッダ，AFM ヘッダを各パケットに要求する．そのため，アプリケーションには，図 5.6に示すパケットフォーマットの利用が要求される．AAF を利用して送信されたデータフローは DM にて処理された後，AFM にて処理される．そのため，図 5.6に示すようなパケットフォーマットとなる．

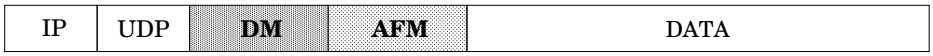

図 5.6: Adaptive Application Flooding がアプリケーションに要求するパケットフォーマット

データフローサイズの削減処理の利用に関するアプリケーションへの要求事項を整理する．AAF はデータフローサイズの削減処理を利用するためにはまず，データサイズの変更可能なデータ構造を持つコンテンツを用意することが必要である．さらに，アプリケーションはコンテンツを用意した上で，パケットサンプリング処理またはデータ間引きのどちらを利用するかで，パケットの送信方法やデータヘッダの記述内容を変更する必要がある．パケットサンプリング処理を利用する場合には，コンテンツを優先度の異なる複数のデータフローに分割して送信すること，最重要な情報を含むデータフローの優先度を高く設定することが必要となる．デー

タ間引き処理を利用するためには，コンテンツのデータ構造を解析するための処理が AFM に拡張モジュールに記述されていること，さらに DM ヘッダ内に利用する処理を指定する必要がある．

5.5 Adaptive Application Flooding がユーザに提供する利点

AAF がデータフローの優先度に応じて提供する機能として，データフローの優先制御機能，データフローサイズの削減機能を示した．本節ではこれらの機能を利用することで，アプリケーションのユーザが得られる利点を示す．

まず AAF を利用するアプリケーションがユーザへ与える利点を説明する．ユーザへの利点としては送信者，受信者それぞれに対するものがある．まず送信者の得られる利点を考察する．送信者にとっての利点とは，ネットワークトラフィックが集中する環境においても，パケットロスや伝送遅延を抑えながらコンテンツの伝搬範囲を拡大出来ることである．これにより，送信者は伝搬範囲内の多くの受信者に対して，再生可能なコンテンツデータを提供することが可能になる．一方で，AAF を利用しない場合は，過度のネットワークトラフィックが発生したときにパケットロス，伝送遅延が増え，受信者に対して再生可能なコンテンツを提供できない．

次に受信者の得られる利点を考察する．受信者の利点とは，複数のコンテンツを受信場所に応じたデータ品質や内容で受信することである．5.3.1節では，送信されるデータフローが伝搬範囲を拡大するにつれて，再送制御やサンプリングの処理対象になりやすいことを示した．これを考慮すると，送信ノードから距離 (ホップ数) が近くなる程，受信者は高い品質のコンテンツを受信するといえる．そのため，受信者はデータ品質の高いコンテンツを再生することで，その送信地点が近隣に存在することを曖昧に知ることが出来る．この特徴はコンテンツ内容が送信ノードの配置される場所の広告情報等である場合，その場所が近隣に配置されていることが分かるといった利点をユーザに提供する．

5.6 本章のまとめ

本章では，過剰なネットワークトラフィックの発生を回避しながらコンテンツフラッディングを実現するアプリケーション基盤として AAF を提案した．次章では AAF の設計を DM，AFM に分けて説明する．

第6章　設計

本章ではAAFの設計を示す．第5章にてAAFを構成するモジュールとしてAFM及びDMを説明した．以降では，AFMとDMに分けてAAFの設計を説明する．

6.1　Autonomous Flow Management moduleの設計

5.3節にて説明したようにAFMの持つ機能は過剰なネットワークトラフィックを検知するための機能，データフロー数やデータフローサイズを削減するための機能の2種類に分類される．AFMは前者の機能を実現するモジュールとしてNetwork Monitoring module，後者の機能を実現するためのモジュールとしてAFM Core module，Data Thinning moduleを持つ．後者の機能を構成するモジュールのうち，データフローの再送制御及びパケットサンプリング処理はAFM Core module，データ間引き処理はData Thinning moduleが行う．図6.1にAFMのモジュール構成を示し，各モジュールの動作概要を説明する．

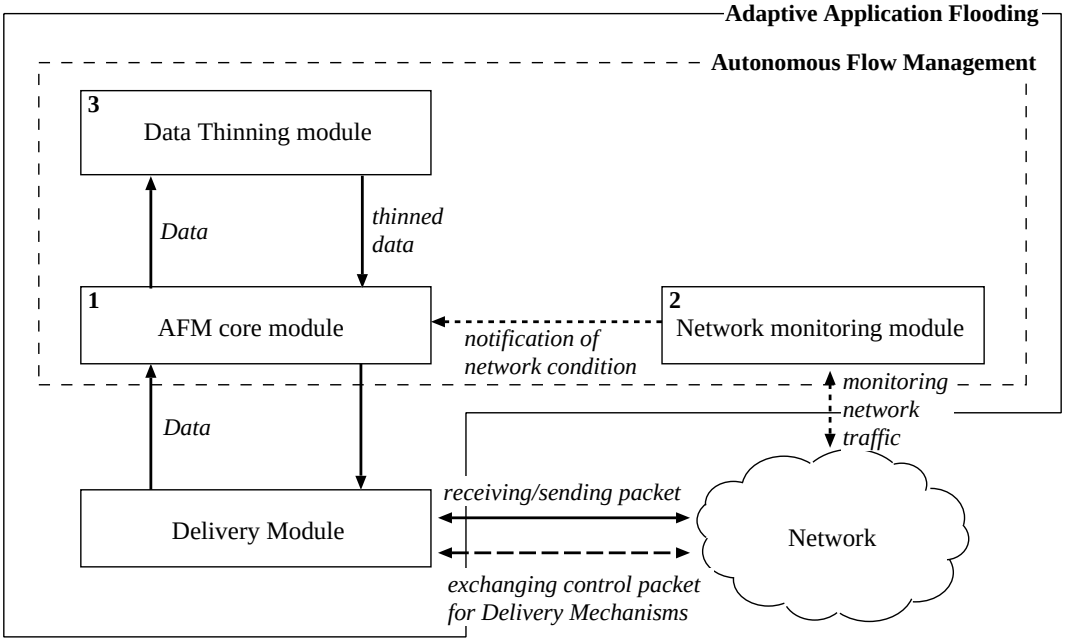

図 6.1: Autonomous Flow Management moduleの設計概要

1. AFM Core module
   AFM Core moduleはAFMを構成するモジュール群をまとめる中心的な役割を持つ．AFM Core moduleはNetwork Monitoring moduleから通知されるネットワーク状態を

基に処理対象とするデータの選定基準 (優先制御ポリシ) を動的に設定する．AFM Core module はこの基準を利用することにより，処理対象とするデータを選定する．選定されたデータはデータ間引き可能なデータである場合に Data Thinning module へデータを渡され，不可能な場合 AFM Core module において再送制御が適用される．また，AFM Core module は AFM を構成するモジュール群の中で唯一，配信機構とデータの送受信を行う．

2. Network Monitoring module
   Network Monitoring module はネットワークトラフィックを常時監視するモジュールである．*Network Monitoring module* はネットワークトラフィックを監視することによりネットワーク状態の変化を検知する．ネットワーク状態の変化を検知した場合は，*AFM Core module* にネットワーク状態を通知する．

3. *Data Thinning module*
   Data Thinning module はデータ間引き処理を行うモジュールである．*Data Thinning module* は *AFM Core module* からデータ間引き処理の可能なデータが渡されると，*DM* ヘッダに指定されるサンプリングレートに応じたデータ間引きを行う．なおデータ間引きには *DM* ヘッダに指定された処理が利用される．データ間引き後，*Data Thinning module* はデータを *AFM Core module* に戻す．

以上，各モジュールの動作概要を説明した．本章では以降，Network Modnitoring module と AFM Core module の連係による優先制御ポリシの動的設定，AFM Core module と Thinning module におけるデータ処理について説明する．

6.2 優先制御ポリシの動的設定

優先制御ポリシは Network Monitoring module の通知するネットワーク状態の変化を基に AFM Core module によって動的に設定される．本節では，Network Monitoring module における処理を説明した後，AFM Core module による優先制御ポリシの設定処理を説明する．

6.2.1 Network Monitoring module によるネットワーク状態の検知

本項では Network Monitoring module がネットワーク状態を検知，通知するための処理を説明する．Network Monitoring module がネットワーク状態の変化を通知するためにはまず，ネットワーク状況を定義する必要がある．ネットワーク状態とは，ネットワークトラフィックの集中，ネットワークトラフィックの安定などを指す．本項ではネットワークトラフィックの集中している状態を CONGESTED，安定している状態を STALE，どちらでもない状態を WARNING として，それぞれの状態を定義した後，ネットワーク状態の遷移条件を説明する．

ネットワーク状態の定義

各ネットワーク状態の定義を行う．まず，CONGESTED は秒毎に発生するトラフィック量が設定する閾値 (Traffic Threshold) を越えた状態と定義する．ここで利用する閾値には，SNR

の低下やパケットロス率が向上しない程度のトラフィック量が設定される．STALE は一定時間 (Polling Time)，状態が CONGESTED に変わらなかった状態と定義する．WARNING は CONGESTED，STALE 以外の状態と定義する．以上定義したように，Network Monitoring module はネットワーク状態の変化を検知するために，閾値として Traffic Threshold，Polling Time の設定を必要とする．

ネットワーク状態の遷移

ネットワーク状態の遷移条件を説明する．Network Monitoring module によるネットワーク状態検知及びネットワークの状態遷移を図 6.2の状態遷移図を利用し，説明する．

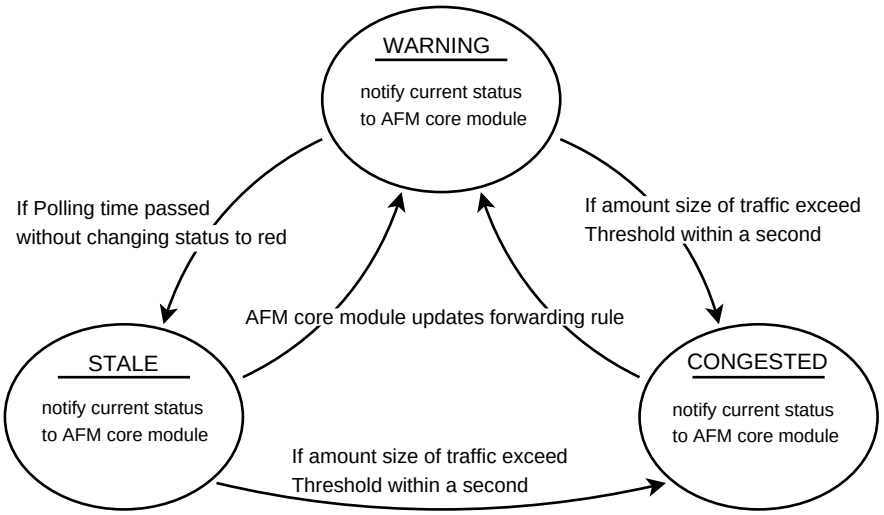

図 6.2: Network Monitoring module において検知されるネットワーク状態遷移

図 6.2に描かれている CONGESTED，WARNING，STALE それぞれの状態間の矢印は状態を遷移するための条件，各状態に記述されている動作はその状態に変わった際に行われる動作を示す．

まず，各状態に遷移した際の動作に注目すると Network Monitoring module は状態遷移後に AFM Core module に対してネットワーク状態を通知していることがわかる．AFM Core module は CONGESTED，WARNING，STALE の 3 種類の通知を受け取ることにより優先制御ポリシを動的に設定する．

次に，状態間の遷移に注目すると，STALE-WARNING 間，WARNING-CONGESTED 間では相互に状態遷移可能であるのに対し，STALE-CONGESTED 間は STALE から WARNING に対してのみ状態遷移可能である．つまり CONGESTED から STALE に状態遷移するためには，必ず WARNING の状態を経由する必要がある．

最後に各状態への状態遷移に注目すると，CONGESTED，STALE，WARNING に状態遷移するためにはそれぞれ，秒毎のトラフィック量が Traffic Threshold を越えること，Polling Time 間に一度も CONGESTED に状態遷移が起こらないこと，AFM Core module が CONGESTED，STALE の通知を優先制御ポリシに適応したことが条件となる．この中でも，WARNING への

状態遷移条件はNetwork Monitoring moduleとAFM Core moduleの間で同期をとるために必要となる．ここで行われる同期は，優先制御ポリシの優先度，Thinning levelがネットワーク状態に準じて設定されるために重要な役割を持つ．

6.2.2 AFM Core moduleによる優先制御ポリシの動的な設定

本項では優先制御ポリシを構成する情報について説明した後，ネットワーク状態通知に対して，AFM Core moduleがどのように優先制御ポリシを設定するか説明する．

まず，優先制御ポリシを構成する情報を示す．AFMは各データの優先度に応じて再送制御やデータフローサイズ削減の対象を決定する．そのため，優先制御ポリシには処理対象を決定するための対象優先度が設定される必要がある．対象優先度を利用することで再送制御やパケットサンプリング処理の対象となるデータフローを決定出来る．優先制御ポリシは，さらに間引き段階 (Target Thinning Level) を各対象優先度毎に設定することで，段階的な間引き処理を実現する．以上示した対象優先度，Target Thinning Levelの2つの情報から優先制御ポリシは構成される．

次に，ネットワーク状態通知に応じてAFM Core moduleがどのように優先制御ポリシを設定するか説明する．優先制御ポリシがネットワーク状態に応じて設定される様子を図6.3に示す．

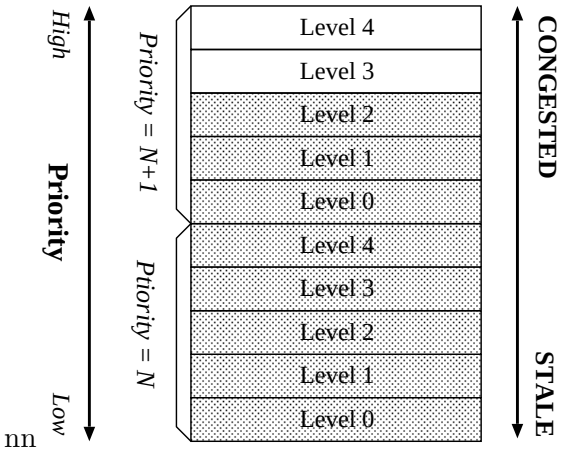

図 6.3: 優先制御ポリシの動的な設定

図6.3に示すように，優先制御ポリシは対象優先度毎に5段階のThinning Levelを持つ．AFM Core moduleはNetwork Monitoring moduleからネットワーク状態に応じてThinning Levelを変化させる．通知内容がCONGESTEDの場合はThinning Levelを上げる，WARNINGの場合は保持する，STALEの場合は下げる処理が行われる．またThinning levelが一定の値を越えると，処理対象となる優先グループが高くなる．この際，Thinning levelは新しく対象となった優先グループに対して低い値からネットワーク状態に応じて適用される．

6.3 各モジュールにおけるデータ処理

AFMはDMから渡されたデータに対して，優先制御ポリシを基に再送制御やパケットサンプリング処理，データ間引き処理が適用する．これにより，AFMはネットワークトラフィックを

抑制する．以降ではまず，各パケットが保持すべき情報群を示す．その後，AFM Core module，Data Thinning module の行う各データへの処理を説明する．

6.3.1 各データの保持する情報群

各データは AFM Core module，Data Thinning module を通じて処理される．従って，AFM Core module，Data Thinning module の処理において必要となる情報を各データは保持すべきである．

まず，AFM Core module の処理に必要となる情報を示す．AFM Core module は優先制御ポリシを利用することで処理対象とするデータを選択する．そのため，データは優先制御ポリシに設定される処理対象優先度，Target Thinning Level に対応した情報が必要である．さらに，データ間引き処理を利用するデータを Data Thinning module に渡すためには，データ間引き処理を適用することが明示される必要がある．

次に，Data Thinning module の処理に必要な情報を整理する．Data Thinning module はデータ間引き処理を担当するモジュールである．5.3.3節に示したようにデータ間引きには，パケット毎のデータ解析処理が必要である．この処理はアプリケーション依存に行われるため，明示的に指定されることが必要となる．そのため，Data Thinning module にはデータ間引き処理の種類を決定するための情報を必要とする．

以上，AFM のデータ処理に関して必要となる情報を挙げた．これらの情報は各パケットの AFM ヘッダに記述される必要がある．AFM ヘッダのフォーマットを図 6.4に示す．

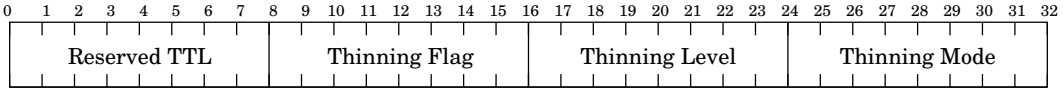

図 6.4: AFM ヘッダのフォーマット

AFM ヘッダは Reserved TTL，Thinning Flag，Thinning Level，Thinning Mode の 4 つのフィールドから構成される．Reserved TTL は優先度，Thinning Flag はデータ間引きの可否，Thinning Level はデータ間引き段階，Thinning Mode はデータ間引き処理を指定するためのフィールドである．このうち Reserved TTL の値は中継ノードを経由するにつれて 1 ずつ減じられる．また，Thinning Level はデータ間引きが一度も行われていない場合は 0 に設定され，データ間引きが行われる毎に 1 ずつ増える．

6.3.2 AFM 全体のデータフロー

6.2.2項にて優先制御ポリシの情報群を，6.3.1項にて AFM ヘッダを定義した．本項では優先制御ポリシ，AFM ヘッダを利用して AFM Core module，Data Thinning module が行うデータ処理の全体像を示す．本項では，図 6.5に示すフロー図を利用して，AFM Core module，Data Thinning module におけるデータ処理を説明する．

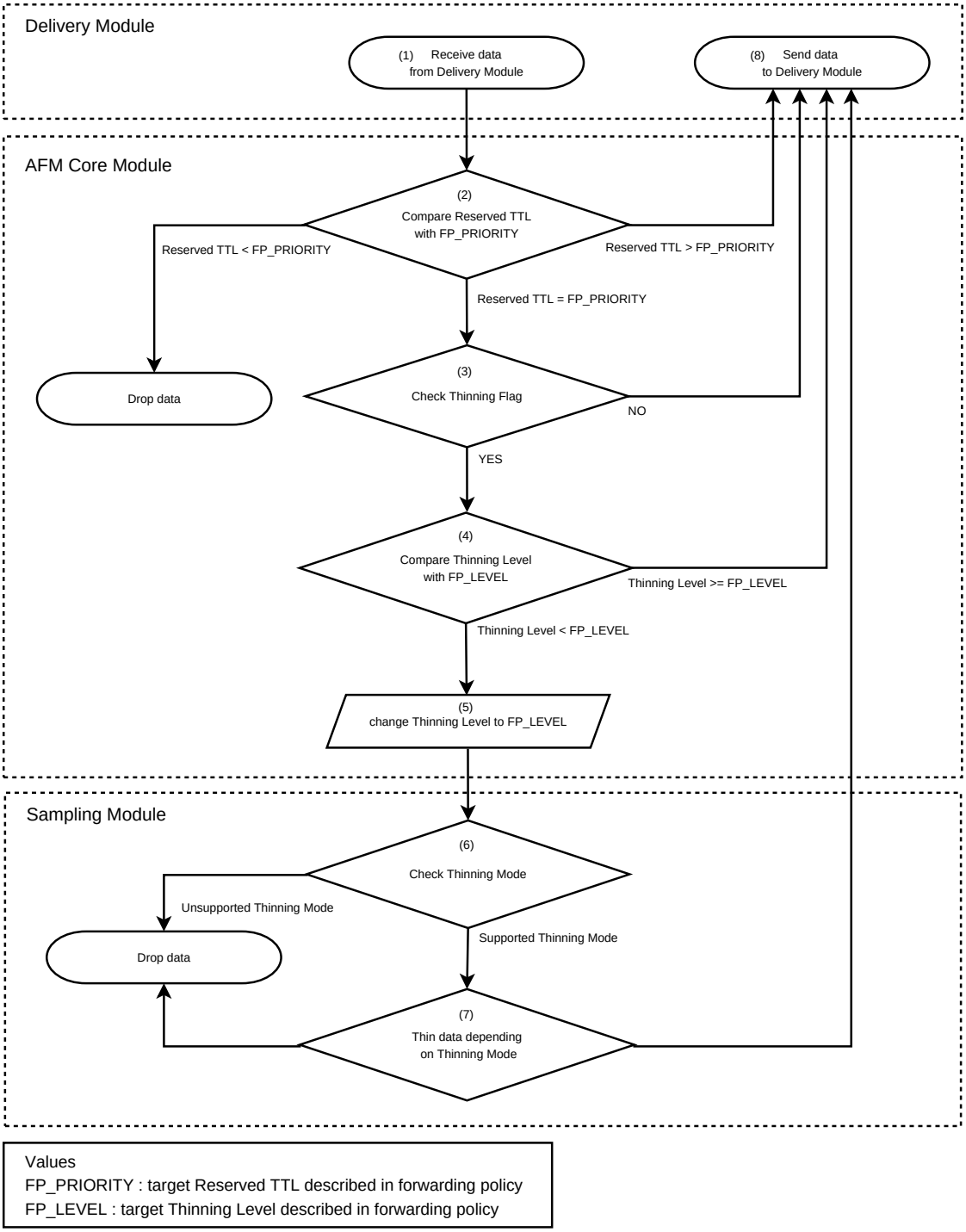

図 6.5: Autonomous Flow Management module におけるデータ処理流れ

1. *Delivery module* からデータが渡される．
2. AFM ヘッダ内の Reserved TTL を優先制御ポリシの処理優先度 (FR_PRIORITY) と比較する．Reserved TTL が FP_PRIORITY よりも高い場合はデータを Delivery module に渡し，低い場合には棄却し，同じ場合には (3) の処理を行う．
3. AFM ヘッダ内の Thinning Flag を参照する．サンプリング不可の場合，データは Delivery module に渡され，可能な場合は (4) の処理を行う．
4. AFM ヘッダ内の Thinning Level を優先制御ポリシの Target Thinning Level(FP_LEVEL) と比較する．Thinning Level が FP_LEVEL より大きい場合，データを Delivery module に渡し，小さい場合は (5) の処理を行う．
5. AFM ヘッダ内の Thinning Level を FP_LEVEL に変更する．その後，データを Data Thinning module に渡す．
6. AFM ヘッダ内の Thinning Mode を参照する．Data Thinning module は Thinning Mode に指定された処理が AFM によりサポートされている場合は (7) の処理を行い，未サポートの場合はデータを棄却する．
7. AFM ヘッダ内の Thinning Mode 及び Thinning Level に対応したデータ間引き処理を行う．
8. *Delivery module* にデータを渡す．

それぞれの処理を *AFM Core module*，*Data Thinning module* に分けて説明する．

まず AFM Core module の処理を説明する．(1)(9) は Delivery module とのデータ送受信処理である．(2) の処理により優先度に応じた再送制御，パケットサンプリング処理が実現される．(3) の処理によりデータ間引きの可否を即座に把握することが出来る．(4) の処理により，AFM Core module の送信するデータの Thinning Level に差が出ることを防ぐことが出来る．(5) の処理は，AFM ヘッダの Thinning Level が Data Thinning module で利用されるために Thinning Level の更新を行っている．これら (2)(3)(4)(5) の処理により，Data Thinning module へデータが渡される．

次に Data Thinning module の処理を説明する．Data Thinning module は AFM Core module からデータを受け取るとまず，(6) の処理により，データ間引き処理を決定する．(7) の処理により，Thinning Level 及び Thinning Mode に基づいたデータ間引き処理が行われる．その後，サンプリングされたデータを AFM Core module に返す．以上が Data Thinning module の動作である．

6.4 Delivery module の設計

AAF を構成する DM の設計に関して説明する．DM にはネットワークトラフィックを抑制するために SMF を利用する．さらに複数のアプリケーションから利用出来るように独自の拡張を加えた．本節では以降，SMF の設計を述べた後，追加した機能について設計を示す．

6.4.1 SMF の設計

DM はネットワークトラフィックを抑制するために SMF を利用する．また MPR を SMF の扱う中継ノードの選択アルゴリズムとして利用する．本項では以降，本研究における SMF の設計を示す.

SMF の設計概要

SMF の設計概要を図 6.6を利用して説明する.

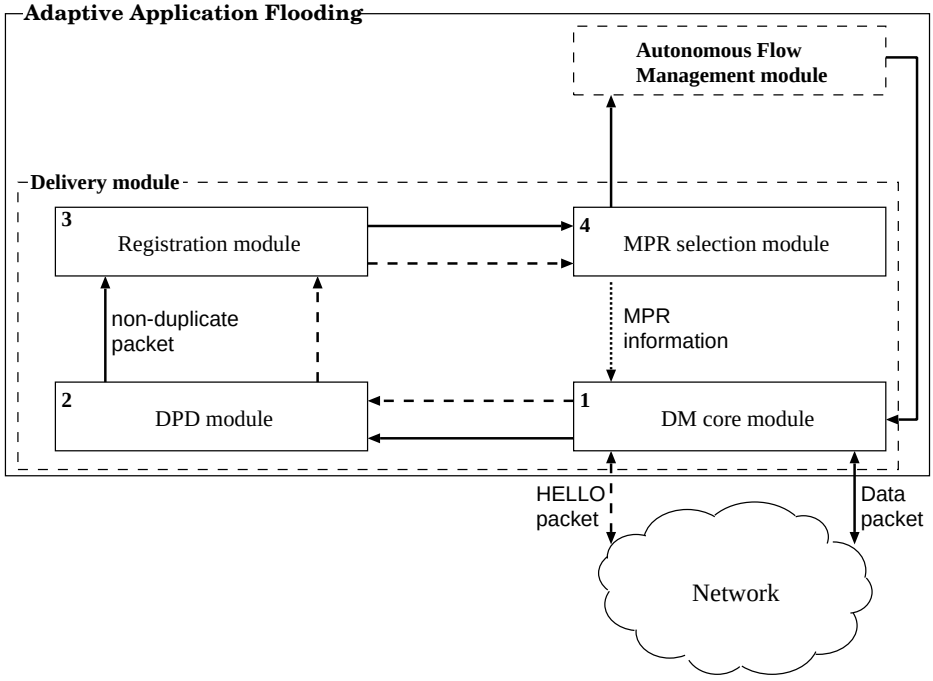

図 6.6: Simplified Multicast Forwarding の設計概要

図 6.6に示すように SMF は DM core module，DPD(Duplicate Packet Detection) module, Registration module，MPR module から構成される．DM core module は他のノードとパケットの送受信を行うだけでなく，他のモジュールとのデータ交換を行う．DPD module は重複パケットの検知を行うモジュールであり，Registration module は DM ヘッダに格納される情報を SMF 内部のレジストリに格納し，管理を行うモジュールである．MPR module は中継ノードの選択を行うモジュールである．図 6.6に示すように，AFM へのデータ渡しは MPR module と DM core module の間で行われる．なお SMF 単体で利用する場合には，MPR module から直接 DM core module へデータが渡される.

以上 SMF の設計概要を説明した．本項では以降，SMF を利用するために必要となるデータヘッダとして DM ヘッダを説明した後，SMF の行うデータ処理を説明する．[1]

[1]SMF の仕様は 2005 年 10 月に更新された Internet draft[9] では，処理に必要な情報を IP ヘッダの拡張フィールドに記述することを定めているが，本論文執筆時点 (2005 年 7 月) における Internet draft[29] では必要な情報の記述部分に関して，議論の段階であった．そのため，本研究における SMF は機能自体は SMF と同様であるものの，IP ヘッダではなくデータヘッダに必要な情報を記述するよう設計した.

DM ヘッダフォーマット

DM ヘッダのフォーマットを図 6.7に示す．DM ヘッダは SMF 利用時に必要となるフィールド以外に，DM が複数のアプリケーションから利用されるために必要となる情報が格納される．

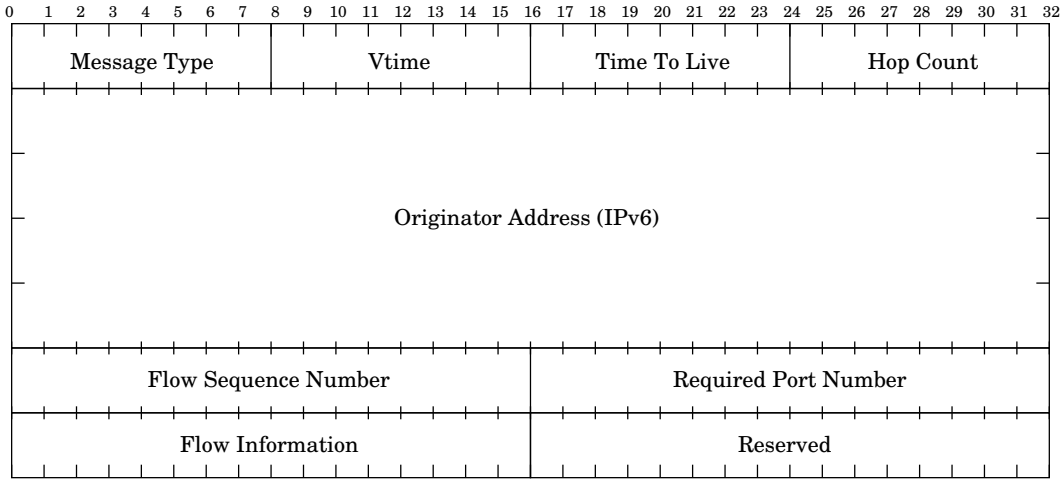

図 6.7: DM ヘッダのフォーマット

DM ヘッダに用意された複数のフィールドの中でも，Time To Live，Hop Count は Originator Address(IPv6) の意味は自明であるため，ここでは他のフィールドに関して説明する．Message Type フィールドはパケット処理を分岐させる役割を持つ．SMF は Message Type フィールドの値により，アプリケーションから渡されるデータの送信，他のノードから渡されたパケットの再送，2 種類の処理を行う．なお，図 6.6のデータの流れは後者の処理が行われたときのものである．それぞれの処理がどのように行われるかは 6.4.1 項に示す．Vtime は Registration module におけるデータの有効時間を示す．Vtime を利用することで 6.4.1項に説明するレジストリから古くなったデータフロー情報のエントリを削除することが可能である．Flow Sequence Number はパケット毎に管理されるシーケンス番号であり，1 パケット毎に 1 ずつ増加する．この値は DPD module における重複パケットの検知に利用される．Required Port Number はユーザが SMF の受信しているコンテンツを，受信アプリケーションの利用する特定のポートに転送するために利用される．Flow Information はユーザに対して，SMF が受信しているコンテンツのフロー毎にテキスト，映像，音声などのデータタイプに加え，ニュースやお勧め音楽など，大まかなコンテンツ内容を通知するために利用される．Required Port Number は 6.4.2項に示す Forwarding Request 機能，Flow Information は 6.4.2項に示す Flow Information Request 機能に利用される．

SMF におけるデータ処理

既に説明したように SMF によるデータ処理は Message Type フィールドの値により 2 種類に分かれる．以降では 2 種類のデータ処理それぞれについて説明する．まず，他のノードからパケットを受信した際のパケットの処理について説明する．パケットは DM core module に受信されると DPD module へと転送され，重複して受信されたパケットでないか調査される．受

信パケットは重複検知のために Registration module に渡され，パケットヘッダの情報は SMF 内のレジストリに登録される．その後，MPR module にてパケットを再送するか否かが決定される．この際，DM ヘッダの Originator Address(IPv6) が参照される．再送する場合は DM Core module にパケットが渡される．以上がパケットを他のノードから受信した際の処理の流れである．次に，ローカルで動作するアプリケーションからデータが渡された場合の処理の流れを図 6.8説明する．

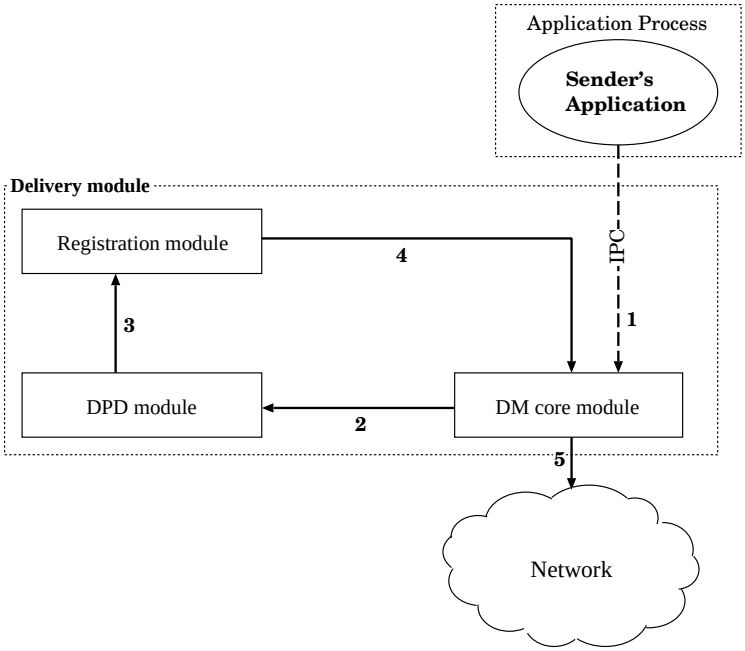

図 6.8: パケット-ネットワーク間のデータ中継機能

DM core module は IPC を通して送信アプリケーションからパケットを受信すると，DPD module に渡し，重複パケットでないか検知した後，Registration module にてヘッダ情報を登録する．その後データは DM core module へと渡され，パケット化され送信される．ローカルで動作するアプリケーションからデータが渡された場合には，MPR module を介すことなくネットワークにパケットが送信される．以上がパケットをアプリケーションプロセスから受信した際のデータ処理である．この処理は DM ヘッダを利用するアプリケーション全てに適用することが出来る．これにより，複数の送信アプリケーションから同時に DM を利用することが出来る．

6.4.2 Delivery module への追加機能

本項では，DM の受信するデータフローを受信アプリケーションに転送する Forwarding Request 機能，DM の受信するデータフロー情報を表示，検索する Information Request 機能を DM に追加した．これらは DM が受ノードにおいて利用される機能である．以降ではそれぞれの機能の設計を示す．

Forwarding Request

Forwarding Request 機能の実現には Flow Request Message がアプリケーションプロセスから DM core module に渡される必要がある．Flow Request Message のメッセージフォーマットを図 6.9に示す.

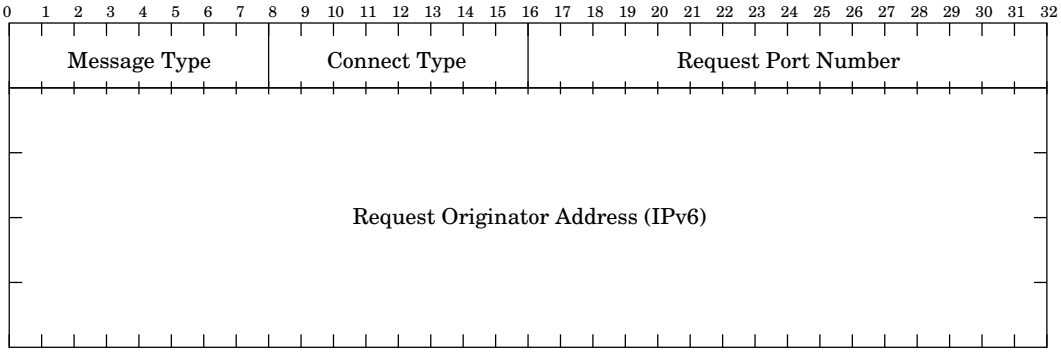

図 6.9: Flow Request Message フォーマット

まず Flow Request Message のメッセージフォーマットに関して説明する．図 6.9に示すように Flow Request Message は Message Type，Connect Type，Request Port Number，Request Originator Addrress(IPv6) の 4 つのフィールドから成る．Message Type は Open，Close の 2 タイプを指定することが出来る．Open はアプリケーションプロセスへのパケットのフォワーディングを開始する場合，Close は終了する場合に利用される．Connect Type は DM-受信アプリケーション間で利用する通信方式を選択するためのフィールドである．このフィールドにより IPC で利用する通信方式として UDP または TCP が選択可能となる．Request Port Number, Request Originator Address は特定のデータフローを DM から受信アプリケーションへフォワーディングする際に利用される．次に，Forwarding Request がどのような手順で行われるかを図 6.10に示す.

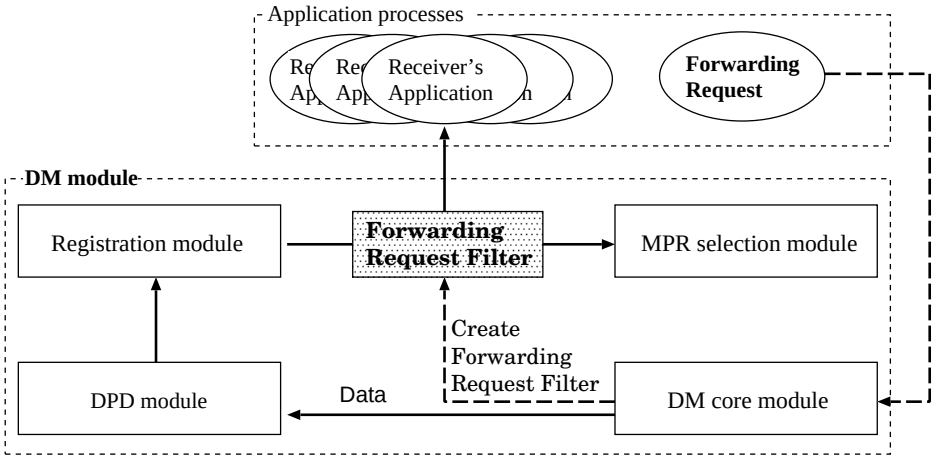

図 6.10: Forwarding Request 機能の処理流れ

Forwarding Request 機能を実現するための処理手順を図6.10を利用し，説明する．Forwarding Request 機能を利用するためには，まずアプリケーションプロセスから Flow Request Message が送信される必要がある．Flow Request Message は IPC を利用し，アプリケーションプロセスから DM core module に渡される．この際，Flow Request Message の Request Port Number, Request Originator Address を指定する必要がある．図6.10に示すように，DM core module は Flow Request Message を受信すると，指定された Originator Address，Request Port Number をヘッダ情報に持つパケットを，受信アプリケーションに転送するために，Forwarding Request Filter を構築する．Forwarding Request Filter は Registration module と MPR module の間に構築され，Registration module から渡されたパケットが指定した Originator Address，Port Number と合致した場合に限り，パケットはローカルホストの指定した Port 番号にフォワーディングされる．

Flow Information Request

6.4.2項にて Forwarding Request 機能を説明したが，Forwarding Request 機能を利用するためには，SMF が現時点で受信しているパケットのヘッダ情報を知る必要がある．Flow Information Request は現時点における SMF のフロー情報をユーザに提供する機能である．Flow Information Request 機能の実現には Information Request Message がアプリケーションプロセスから DM core module に送信される必要がある．Information Request Message のメッセージフォーマットを図 6.11に示す．

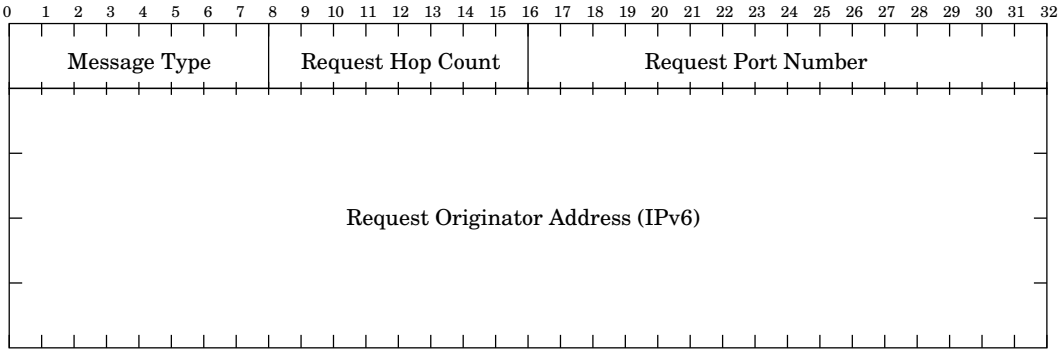

図 6.11: Information Request Message フォーマット

図 6.11に示すように Information Request Message は Message Type，Request Hop Count, Request Port Number，Request Originator Address(IPv6) の 4 つのフィールドを持つ．Message Type を変更することにより，全ての Flow 情報，指定した Originator Address に合致する Flow 情報，指定した Port 番号に合致する Flow 情報，指定した Hop Count に合致する Flow 情報，これらのどれを取得するかが決まる．Request Hop Count 及び Request Port Number, Request Originator Address はこれらの Message Type に応じて利用される．Information Request Message が DM core module に渡されると DM core module は SMF 内の Flow 情報を管理するレジストリにアクセスする．この際レジストリからは Flow 毎に Originator Address 及び Required Port Number，Flow Information，3 つの情報が取得される．DM core module は

指定された条件に合致する Flow 情報を取得すると，その結果をアプリケーションプロセスへ渡す.

6.5 本章のまとめ

本章では AAF を構成するモジュールとして AFM 及び DM の設計を示した．次章では AAF の実装を示す.

第7章　実装

本章ではAFM及びDMの実装を示す.

7.1　実装概要

本研究ではAAFの配信機構としてDMをフローコントロール機構としてAFMを実装した.また，AAFを複数のアプリケーションから利用可能となるように，独自の機能をDMに追加した．本研究においてAFMやDMに実装した機能を以下に示す.

- Autonomous Management module
  - ネットワーク状態の把握
  - 優先制御ポリシの動的設定
  - 優先制御ポリシに応じた再送制御
  - 優先制御ポリシに応じたデータフローサイズ削減処理
- Delivery module
  - Simplified Multicast Forwarding
  - 送信アプリケーションからネットワークへのデータ中継機能
  - ネットワークから受信アプリケーションへのデータ転送機能
  - Delivery moduleの受信するデータフロー情報の表示・検索機能

7.1.1　実装環境

本実装の実装環境を表7.1に示す.

表 7.1: 実装環境

使用OS	Debian GNU/Linux kernel-2.6.9
使用言語	C言語
使用ライブラリ	zebra 0.95-pre2, libpthread, libpcap

- GNU Zebra
  GNU ZebraはGNU GPL(GNU Public Licence)に基づいて開発されたオープンソースな経路制御パッケージであ．GNU Zebraは，BGP4，RIPv1，RIPv2，OSPFv2をサポー

トしており，各種 UNIX 系 OS 上で動作する．また，GNU Zebra は，経路制御デーモンの開発用ライブラリを持つことで，経路制御プロトコルの開発環境を提供する．本実装では，GNU Zebra のライブラリを利用することで，SMF の実装を行った．

- libpthread
  libpthread は POSIX 準拠の thread ライブラリである．libpthread を利用することにより，マルチスレッドで動作するプログラムが実装可能である．

- libpcap
  libpcap はユーザレベルでのパケットキャプチャを OS 非依存に実現する，オープンソースなパッケージである．libpcap はデータリンクレベルでのネットワーク監視に対し，汎用的なフレームワークを提供する．本実装では，libpcap を利用することにより，AFM の実装を行った．

7.1.2 Adaptive Application Flooding のコマンドラインオプション

Adaptive Application Flooding で利用可能なオプションを表 7.1に示す．

```
-d, --daemon          Runs in daemon mode
-f, --config_file     Set configuration file name
-i, --interface       Set using interface
-n, --no_relay        Not relay any packet
-p, --play            Deliver data to playback process
-a, --with_afm        Use AFM (SMF extention module)
-t, --threshold       Set Traffic Threshold for AFM
-T, --polling_time    Set Polling Time for AFM
-h, --help            Display this help and exit
```

図 7.1: Adaptive Application Flooding のコマンドラインオプション

これらのオプションのうち，Adaptive Application Flooding に関連するオプションを説明する．まず，-n オプションは，データフローの受信処理のみを行う場合に利用される．-p オプションは，ネットワークから受信アプリケーションへのデータ転送機能を有効化するために利用される．-a オプションは，AFM を有効化するために利用される．-t オプションは，Traffic Threshold を設定するために，-T オプションは Polling Time を設定するために利用される．

7.2 Autonomous Flow Management module の実装

AFM module は AFM Core module，Network Monitoring module，Data Thinning module，これら 3 つのモジュールから構成される．これら 3 つのモジュールに対して実装した機能を表 7.2に示す．

表 7.2: AFM の各モジュールに実装された機能

実装した機能	対応するモジュール
ネットワーク状態の把握	Network Monitoring module
優先制御ポリシの動的設定	AFM Core module, Network Monitoring module
データフローの再送制御	AFM Core module
データフローサイズ削減処理	AFM Core module，Data Thinning module

7.2.1 優先制御ポリシの動的設定

優先制御ポリシの動的設定に関して実装内容を示す．本研究では，優先制御ポリシの動的設定を実現するために，以下の処理を実装した．

1. Network Monitoring module によるネットワークトラフィックの常時監視
2. Network Monitoring module によるネットワーク状態の検知
3. Network Monitoring module から AFM Core module へのネットワーク状態の通知
4. Network Monitoring module - AFM Core module 間でのネットワーク状態の同期

本項では以降，これらの機能が本実装においてどのように実現されているか，説明する．また 3，4 点目に示した機能に関しては，ネットワーク状態の共有を行うための機能であるため，まとめて説明を行う．

ネットワークトラフィックの常時監視

本実装では，ネットワークトラフィックの常時監視を実現するために，プロミスキャスモードを利用した．さらにプロミスキャスモードを有効にする方法として，libpcap を利用した．プロミスキャスモードを有効にする方法は OS 毎に異なる．そのため OS 毎に異なる実装が必要となるが，本実装では libpcap を利用することにより OS 非依存にプロミスキャスモードを有効化する．libpcap は OS 依存部分を隠蔽化することで，多くの OS から利用可能な API を提供する．

ネットワーク状態の検知

ネットワークトラフィックの流量は連続的であるため，そのままでは特定のネットワーク状態を検知出来ない．第 6章では，ネットワークの状態として CONGESTED，WARNING，STALE，これら 3 つのネットワーク状態の定義，及び定義に必要となる指標として Traffic Threshold，Polling Time，2 つの指標を示した．本実装では，これらの指標を afm_thresholds 構造体に格納して利用する．図 7.2に示すように，afm_thresholds 構造体は Traffic Threshold として cns_threshold，Polling Time として cns_interval を利用した．本実装では，これらの数値をプログラム起動時に設定する必要がある．

Network Monitoring module は，afm_thresholds 構造体，プロミスキャスモードの利用により監視されるネットワークトラフィック流量を利用することで，ネットワーク状態の遷移を検

```
struct afm_thresholds
{
  u_int16_t cns_interval;
  u_int32_t cns_threshold;
}
```

図 7.2: afm_thresholds 構造体

知する．本実装で利用した，ネットワーク状態遷移を検知するためのアルゴリズムを図 7.3に示す.

- ネットワークトラフィック流量の監視 (1 行目，5 行目)
  ネットワーク状態遷移を検知するアルゴリズムは，libpcap の callback 関数内に定義した．callback 関数は，ノードがイーサネットフレームを受信する毎に呼び出される関数である．callback 関数の引数である h はイーサネットフレームのヘッダを格納するための構造体である．本実装ではこの構造体に格納されるイーサネットフレーム長を，変数 sum に計上することにより，ネットワークトラフィック流量を監視する．

- CONGESTED へのネットワーク状態遷移 (3-5 行目, 28-35 行目)
  CONGESTED 状態への遷移は START_T_TIMER()，STOP_T_TIMER()，sum を利用し検知される．3-4 行目にて，フレームの受信時に sum が 0 である場合に，START_T_TIMER() が実行される．また 29 行目にて，sum が Traffic Threshold を超えた場合に STOP_T_TIMER() が実行される．CONGESTED 状態への遷移は，この START_T_TIMER()，STOP_T_TIMER() の差分が 1 秒より少ない場合に起こる．また，Traffic Threshold を超えた場合，sum には 0 が代入される．

- CONGESTED から WARNING へのネットワーク状態遷移 (10-13 行目)
  CONGESTED から WARNING への状態遷移は，Network Monitoring module - AFM Core module 間でネットワーク状態が共有されていること，及びネットワーク状態が CONGESTED であることを条件に起こる．また，CONGESTED への状態遷移時に START_P_TIMER() が実行される．

- WARNING から STALE へのネットワーク状態遷移 (14-18 行目)
  WARNING から STALE への状態遷移は，Network Monitoring module - AFM Core module 間でネットワーク状態が共有されていること，及びネットワーク状態が WARNING であることが必要である．さらに，15 行目で実行される STOP_P_TIMER() と，WARNING 状態への遷移時に実行される START_P_TIMER() の差分が，Polling Time を超えることが必要である．

- STALE から WARNING へのネットワーク状態遷移 (19-22 行目)
  STALE から WARNING への状態遷移は，Network Monitoring module - AFM Core module 間でネットワーク状態が共有されていること，及びネットワーク状態が STALE であることを条件に起こる．

```
1  void check_network_status(userdata, h, p)
2     {
3      if (sum == 0)
4        START_T_TIMER();
5      sum += h->len; /* add frame size */
6      if (update_policy != cns.update_policy){
7          update_policy = cns.update_policy;
8          switch (cns.status){
10      case CONGESTED:
11             cns.status = WARNING;
12             START_P_TIMER();
13             break;
14           case WARNING:
15             STOP_P_TIMER();
16             if (DIFF_P_TIMER > thresholds.cns_interval)
17               cns.status = STALE;
18             break;
19           case STALE:
20             cns.status = WARNING;
21             START_P_TIMER();
22             break;
23           default:
24             break;
25         }
26       cns.update_status++;
27     }
28     if (sum > thresholds.cns_threshold){
29       STOP_T_TIMER();
30       if (DIFF_TIMER() < 1){
31         cns.status = CONGESTED;
32         cns.update_status++;
33       }
34       sum = 0;
35     }
36   }
```

図 7.3: ネットワーク状態遷移の検知アルゴリズム

以上，ネットワーク状態の遷移を検知するアルゴリズムを説明した．これらの状態遷移を実現するために，Network Monitoring module - AFM Core module 間でのネットワーク状態の共有が必要であることを示した．本実装において，ネットワーク状態の共有がいかにして行わ

れるか，次に示す．

Network Monitoring module - AFM Core module 間でのネットワーク状態の共有

本実装では，Network Monitoring module - AFM Core module 間でネットワーク状態を共有するために，スレッド間通信を利用した．スレッド間通信には，libpthread を利用した．スレッド間通信以外にプロセス間通信を利用することも出来る．しかし，ネットワーク状態の共有がイーサネットフレームの受信毎に行われることを考慮すると，プロセス間通信ではオーバーヘッドが高くなる．

本実装では，Network Monitoring module - AFM Core module 間で共有される情報として，ネットワーク状態以外に，Network Monitoring module がネットワーク状態遷移を検知したことを示す update_status，AFM Core module がネットワーク状態をポリシに反映したことを示す update_policy を利用する．これらの情報を格納する構造体を図 7.4に示す．

```
struct cns_set
{
  u_int8_t update_status;  /* notification of status change */
  u_int8_t update_policy;  /* notification of update policy */
  u_int8_t status;         /* current status of network traffic */
};
```

図 7.4: cns_set 構造体

図 7.3に示したように，status には CONGESTED，WARNING，STALE の情報が格納される．update_status に注目すると，図 7.3の 26 行目，32 行目において値が変更されていることがわかる．Network Monitoring module からのネットワーク状態通知は，AFM Core module が，updata_status をデータ受信時に毎回参照することにより行われる．ネットワーク状態の同期は，AFM Core module が，このネットワーク状態通知に対して，ポリシ変更通知を行うことによって実現される．またポリシ変更通知の際，AFM Core module は優先制御ポリシの変更処理を行う．

次に，この優先制御ポリシに関して説明する．6.2節に示したように，優先制御ポリシは，処理優先度，Target Thinning Level の 2 つの情報により構成される．本実装では，これらの情報を格納する構造体として，afm_status_set 構造体を利用する．afm_status_set 構造体は，処理優先度として target_group，Target Thinning Level として target_level を持つ．

また優先制御ポリシは，各処理優先度に対して 5 段階の Target Thinning Level を持つ．図 7.6は，AFM Core module が，Network Monitoring module からのネットワーク状態通知に対して，どのように優先制御ポリシを変更するかを示している．

- CONGESTED 状態の通知に対する処理 (3-9 行目)
  AFM Core module は CONGESTED 状態が通知されると，target_level が MAX_THINNING_LEVEL を超えていないか，検査する．MAX_THINNING_LEVEL より低い場合には target_level

```
struct afm_status_set
{
  int target_level; /* target thinning level of this group */
  int target_group; /* target group for processing */
};
```

図 7.5: afm_status_set 構造体

```
1  /* struct afm_status_set *ss; */
2  switch(cns.status){
3    case CONGESTED:
4      if (ass->target_level < MAX_THINNING_LEVEL)
5        ass->target_level++;
6      else if (ass->target_level >= MAX_THINNING_LEVEL){
7        ass->target_level = 0;
8        ass->target_group++;
9      }
10     update_policy++;
11     break;
12   case STALE:
13     if (ass->target_level > 0)
14       ass->target_level--;
15     else if (ass->target_level == 0){
16         if (ass->target_group > 0){
17             ass->target_level = MAX_THINNING_LEVEL;
18             ass->target_group--;
19         }
20     }
21     update_policy++;
22     break;
23   case WARNING:
24     update_policy++;
25     break;
26   default:
27     break;
28   }
```

図 7.6: AFM Core module による優先制御ポリシの動的設定

を上げ，それ以外の場合には target_level を 0 にして，優先制御の処理対象とする target_group を下げる．AFM Core module は，これらの処理の後，ポリシ変更通知として update_policy の値を変更する．

- WARNING 状態の通知に対する処理 (23-25 行目)
  AFM Core module は WARNING 状態に対して，update_policy の変更のみを行う．

- STALE 状態の通知に対する処理 (12-22 行目)
  AFM Core module は STALE 状態が通知されると，まず target_level が 0 であるか調べる．target_level が 0 より高い場合は，target_level を下げる．target_level が 0 の場合は，target_group が設定されていることを条件に，target_level を MAX_TARGET_LEVEL に変更し，制御する target_group を下げる．AFM Core module は，これらの処理の後，update_policy の値を変更する．

以上，Network Monitoring module - AFM Core module 間におけるネットワーク状態の共有及び，優先制御ポリシの動的設定に関して実装内容を示した．本節では以降，優先制御ポリシに応じたデータフローの再送制御，データフローサイズ削減処理に関して実装内容を示す．

7.2.2 AFM Core module による優先制御ポリシに応じたデータフローの再送制御

本実装では，優先制御ポリシに応じたデータ処理を実現するために，6.3に示した AFM ヘッダを利用する．図 7.7に本実装で AFM ヘッダとして利用した afm_header 構造体を示す．

```
struct afm_header
{
  u_int8_t reserved_ttl;
  u_int8_t thinning_flag;
  u_int8_t thinning_level;
  u_int8_t thinning_mode;
};
```

図 7.7: afm_header 構造体

以降，この afm_header 構造体を利用した，データフローの再送制御やデータフローサイズ削減処理の対象となるデータフローを決定するために，AFM Core module が行う動作を図 7.8に示す．

- 再送制御するデータフローの決定 (1-2 行目)
  再送制御するデータフローは，AFM ヘッダ内の reserved_ttl と target_group の値を比較することにより，決定される．reserved_ttl が処理対象の target_group の値より高い場合には，そのデータは再送制御の対象となる．また，この処理を利用することで，パケットサンプリング処理が実現出来る．

```
1  if (ah->reserved_ttl > ass->target_group)
2    act = ACT_DISCARD;
3  if (ah->reserved_ttl == ass->target_group){
4    if (ah->thinning_flag){
5      if (ah->thinning_level < ass->target_level){
6        act = ACT_THINNING;
7        ah->thinning_level = ass->target_level;
8      }
9    }
10 }
```

図 7.8: AFM Core module による優先制御ポリシに応じたデータ処理決定

- Data Thinning module へ渡すデータフローの決定 (3-10 行目)
  データ間引きの対象となるデータフローは，AFM ヘッダ内の reserved_ttl，thinning_flag，thinning_level の値により決定される．reserved_ttl，thinning_flag により，サンプリング可能であることが検知されると，次に thinning_level が参照される．この thinning_level が，優先制御ポリシに記述される target_level より低ければサンプリング対象に決定される．またこの際，AFM ヘッダ内の thinning_level が更新される．

以上，再送制御やデータ間引き処理の対象となるデータフローの決定アルゴリズムを示した．この後，再送制御の対象となったデータフローは棄却され，データ間引き処理の対象となったデータフローは Data Thinning module に渡される．次に Data Thinning module にて行われる処理を示す．

Data Thinning module によるデータ間引き処理

データ間引き処理は，AFM ヘッダ内の target_level や thinning_mode，さらに thinning_mode に指定される処理が Data Thinning module に記述される必要がある．第 6章にて説明したように，データ間引き処理はアプリケーション依存の処理であり，アプリケーション毎に実装する必要がある．本研究ではこれらの間引き処理に関しては実装対象外とするが，データ間引き処理の欠点を考慮して Data Thinning module を実装した．

データ間引き処理の欠点とは処理時間を必要とすることである．処理時間の増加はパケット毎にデータを解析することや，間引いたデータが一定以上バッファに溜ることでパケットを再構成・送信するようなアプリケーションが利用されることにより生じる．データ間引き処理はこの欠点により，他のデータフローの処理に遅延を発生させる可能性がある．そのため，本研究では Data Thinning module をスレッドとして実装し，AFM Core module とのデータ交換にプロセス間通信を利用した．

7.3 Delivery moduleの実装

DM はネットワークトラフィックを抑制するために SMF の機能，さらに複数のアプリケーションから同時利用できるように Forwarding Request 機能，Flow Information Request 機能を追加した．以降ではそれぞれの実装内容を示す．

7.3.1 Simplified Multicast Forwardingの実装

6.4.1にて示したように，送信アプリケーションが SMF を利用するためには，アプリケーションの送信するデータに DM ヘッダが付随される必要がある．本実装では，図 7.9に示す dm_data_header を DM ヘッダとして利用する．また，アプリケーションが DM を利用してコンテンツ配信するためには，プロセス間通信を用いて DM にデータを送信する必要がある．本実装ではプロセス間通信に UNIX ドメインソケットを利用する．

```
struct dm_data_header
{
  u_int8_t type;
  u_int8_t vtime;
  u_int8_t ttl;
  u_int8_t hopcount;
  struct in6_addr originator;
  u_int16_t seq;
  u_int16_t required_port;
  u_int16_t flow_info;
  u_int16_t reserved;
};
```

図 7.9: dm_data_header 構造体

7.3.2 Delivery moduleへの追加機能

Adaptive Application Flooding は，複数のアプリケーションから利用されることを想定している．そのため，Adaptive Application Flooding には，複数のアプリケーションから利用出来る API の実装が必要である．本実装では，Delivery module である Simplified Multicast Forwarding(SMF) にアプリケーションミドルウェアの機能を持たせるための API を実装した．SMF に追加した実装を以下に示す．

7.3.3 Forwarding Request

本実装では，Forwarding Request を実現するために図 7.10に示す forwarding_request_message, dm_data_header を利用する．Forwarding Request は，dm_data_header に記述される Origina-

表 7.3: Delivery Module に追加した機能

追加した機能	内容
Forwarding Request	ネットワークから受信アプリケーションへのデータ転送
Flow Information Request	DM の受信するデータフローの情報表示

tor Address(IPv6)，required_port の情報を利用することにより，指定する受信アプリケーションへのデータ転送を行う．

```
#define FORWARDING_START 0
#define FORWARDING_END 1

struct forwarding_request_message
{
  u_int8_t type;
  u_int8_t connect_type;
  u_int16_t request_port;
  struct in6_addr request_addr;
}
```

図 7.10: forwarding_request_message 構造体

DM Core module は forwarding_request_message を受信すると，その内容に基づき，受信アプリケーションへデータ転送するための Forwarding Request Filter(FRF) を設定・解除する．図 7.11に示す forwarding_request_set 構造体は，FRF として機能する．FRF を設定・解除するためには，メッセージタイプ，再生するデータフローの送信者アドレス，通信ポート番号，さらに通信方式を指定する必要がある．connect_type の値により DM-受信アプリケーション間で利用される通信方式に UDP，TCP どちらを利用するか決定出来る．DM は，これらの情報を forwarding_request_set 構造体に格納した後，指定された通信ポート番号を利用し，通信ソケットの設定を行う．

以上，forwarding_request_message により，FRF が設定されることを説明した．以下に示す図 7.12は，FRF により受信アプリケーションに転送するデータフローが決定する処理を示す．

図 7.12に示した関数 dm_forwarding_request_set_lookup() は，パケット受信毎に行われる．この関数は，DM ヘッダに記述されるアドレス，ポート番号を引数に取り，これらが forwarding_request_set に合致するかを調べる．合致した場合には，合致した forwarding_request_set が返される．図 7.11にも示したように，この構造体には，通信時に必要となるソケットなどの情報も格納されている．これらの情報を利用することにより，データフローの要求する受信アプリケーションにデータを転送出来る．

以上，Forwarding Request 機能の実装内容を示した．この機能を利用することにより，複数のアプリケーションを同時に利用することが出来る．この機能を利用するためには，DM の受信するデータフロー情報を受信者が把握している必要がある．次に，DM の受信するデータフ

```
struct forwarding_request_set
{
  int P_sock;
  u_int8_t P_connect_type;
  u_int16_t P_request_port;
  struct in6_addr P_request_addr;
  struct sockaddr_in6 P_sin6
};
```

図 7.11: forwarding_request_set 構造体

```
1  struct forwarding_request_set *
2  dm_forwarding_request_set_lookup(struct in6_addr *addr, u_int16_t port)
3  {
4    for(node = listhead(dm.forwarding_request_set); node; nextnode(node))
5      {
6        frs = (struct forwarding_request_set *) node->data;
7        if (!memcmp(addr, &frs->P_request_addr, sizeof(struct in6_addr)))
8          if (port == frs->P_request_port)
9            return frs;
10     }
11   return NULL:
12 }
```

図 7.12: 受信アプリケーションに転送するデータフローの決定

ロー情報の表示・検索機能である，Flow Information Request の実装内容を示す.

7.3.4 Flow Information Request

本実装では，アプリケーションプロセスから Flow Information Request を利用するためのメッセージとして，図 7.13に示す，info_request_message を利用する．info_request_message に，ホップ数やポート番号，送信ノードのアドレスを指定することにより，特定のデータフロー情報のみを取得することが出来る．また，info_request_message に記述される type は，これらの検索条件を決定するために用いられる.

DM Core module は info_request_message を受信すると，Registration module に管理されるデータフロー情報を参照する．その際，info_request_message の type，及び対応する検索条件を利用して，データフロー情報の抽出が行われる．以下の図 7.14は Flow Information Request を利用した，サンプルアプリケーションを利用して得られた出力を示す.

```
struct info_request_message
{
  u_int8_t type;
  u_int8_t hop;
  u_int16_t port;
  struct in6_addr originator;
};
```

図 7.13: info_request_message

```
select print out type (all|originator|port|hop): all
Origiantor: [fec0::3] Hop count: (min)2 (max)2
        Flow[1]: Request Port:  8000 Information:   0
        Flow[2]: Request Port:  7000 Information:   0
        Flow[3]: Request Port:  6000 Information:   0
Origiantor: [fec0::1] Hop count: (min)1 (max)1
        Flow[1]: Request Port:  8000 Information:   0
        Flow[2]: Request Port:  8500 Information:   0
```

図 7.14: Flow Information Request によるデータフロー情報の表示

図 7.14に示すように，データフロー情報は送信ノード毎に出力される．また，Hop count に関して，min，max の 2 つが出力が行われている．min，min はそれぞれ，送信ノードの送信する複数のデータフローの中で，最もホップ数の低いデータフロー，高いデータフローを示す．この 2 つの値の差分が大きい場合，送信ノードまでのネットワークに，複数のノードが密集していることが分かる．Information には AFM ヘッダに記述される Flow Informwation フィールドの値が表示される．ユーザはこの数値に応じてニュースや広告情報といったコンテンツの大まかな内容を把握することが出来る．図 7.14の出力結果に示すように，Flow Information Request を利用することで，受信者がデータフロー情報を取得できる．

7.4 本章のまとめ

以上，AAF を構成する DM 及び AFM の実装を説明した．次章では AAF によりネットワークトラフィックがどのように抑制されるか評価する．

第8章　評価

本章では本研究の提案手法である AAF の評価を行う．

8.1　評価指針

第 2章で提案したサービスモデルを展開するためには，ネットワークトラフィックの抑制が必要である．本章では，第 5章にて提案した AAF がこの目的を達成しているか評価する．AAF はネットワークトラフィックを抑制するために中継ノードの選択処理やデータフローサイズ削減処理を行うが，本研究ではこのうち中継ノードの選択処理に関しては既存の手法である MPR を利用した．MPR に関してはこれまでに多くの評価が行われており，中継ノード選択処理に関しては有意性が認められている [8][16]．一方で，データフローサイズの削減処理に関しては本研究の提案した内容であり，評価する必要がある．本章ではデータフローサイズ削減処理として以下の 2 点が達成されているか実験により評価を行った．

- ネットワーク状態に応じたネットワークトラフィックの抑制
- 優先度や Target Thinning Level に応じたデータフローサイズ削減処理

8.2　評価環境の構築

評価環境はアドホックネットワーク環境をエミュレーションを利用することで構築した．この理由は，実際の無線環境を利用する場合には，無線範囲や無線強度といった様々な要因が AFM の評価に影響を及ぼす可能性があるためである．以降では評価環境をどのように構築したか説明する．はじめに，評価実験を行うために利用した端末を図 8.1に示す．

表 8.1に示すように，実験には 10 台の端末をアドホックネットワークにおけるノード，1 台の端末をエミュレータとして利用した．エミュレータは 10 個のイーサネットポートを持っており，各ノードはそれぞれ，このイーサネットポートに有線で接続する．表 8.1において各ノードの IPv6 アドレスの横に示すインタフェース名は，それぞれのノードがエミュレータのどのイーサネットポートに接続しているかを示す．次にこれらの機材がどのように接続されるかを図 8.1に，エミュレートするネットワークトポロジを図 8.2に示す．

図 8.2に示すように，エミュレートするネットワークトポロジはノード R にネットワークトラフィックが全て集まるようになっている．これはノード R の地点において意図的に過剰なネットワークトラフィックを発生させることで，AFM の挙動を確認するためである．また，エミュレートするネットワークではノードの送信するパケットは実線で結ばれた隣接ノード全てに受信される．

次に，ネットワークトポロジをエミュレートする手順を説明する．アドホックネットワークのトポロジをエミュレートするためには，まずエミュレータが全てのイーサネットポートをブ

表 8.1: 実験機材

Node	MAC Address	OS	CPU	MEM
fec0::1(eth0)	00:14:51:17:cf:48	Mac OSX darwin kernel-8.2.0	PowerPC G4 1.5GHz	512MB
fec0::2(eth1)	00:14:51:17:d0:a4	Mac OSX darwin kernel-8.2.0	PowerPC G4 1.5GHz	512MB
fec0::3(eth2)	00:11:24:d0:d1:32	Mac OSX darwin kernel-8.2.0	PowerPC G4 1.5GHz	512MB
fec0::4(eth3)	00:11:24:d0:cf:90	Mac OSX darwin kernel-8.2.0	PowerPC G4 1.5GHz	512MB
fec0::5(eth4)	00:0d:60:77:fc:39	Debian Linux kernel-2.6.9	PentiumM 1.4GHz	256MB
fec0::6(eth5)	00:09:6b:a0:a6:71	Cent OS kernel-2.6.1	PentiumIII 1066MHz	631MB
fec0::7(eth6)	00:0d:60:78:c6:b3	FreeBSD-5.4	PentiumM 1.4GHz	256MB
fec0::8(eth7)	00:01:80:5c:0a:cf	Debian Linux kernel-2.6.9	PentiumM 1.7GHz	1024MB
fec0::9(eth8)	00:0d:60:7f:22:60	FreeBSD-5.4	PentiumM 1.4GHz	256MB
fec0::a(eth9)	00:0b:5d:2c:27:a9	Cent OS kernel-2.6.1	PentiumM 1GHz	512MB
emulator		Cent OS kernel-2.6.1	Xeon 1.8GHz	1024MB

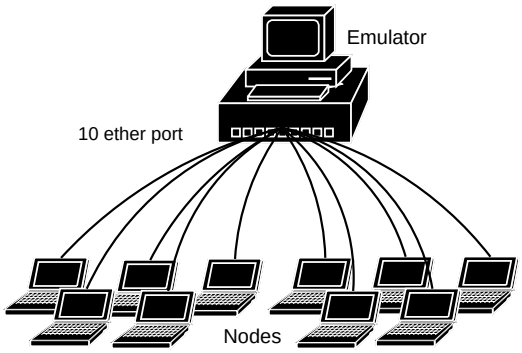

図 8.1: 実際のネットワークトポロジ

リッジングする必要がある．これにより，エミュレータはハブと同様の動作をするようになる．従って，あるノードの送信したパケットは他のノードのネットワークインタフェースまで届くようになる．トポロジをエミュレートするためには，ブリッジングを有効にした状態で，あるポートに対して指定された MAC アドレス以外からはデータを流さないように MAC アドレスフィルタリングを行う必要がある．実験環境の構築には，エミュレータが Linux 端末であったため，ブリッジの設定を行う brctl と，MAC アドレスフィルタリングを行う ebtables を利用した．実験で利用した，ebtables によるフィルタリングルールを図 8.3に示す．

図 8.3に示すように，ソース (-s) となる MAC アドレスと出力するポート (-o) の指定を行うことで，指定した MAC アドレスのみを指定したポートにフォワーディングすることで，図 8.2に示したネットワークトポロジをエミュレートすることが出来る．なお，右の#でコメントアウトされている数字は各ノード IPv6 アドレスの下位の数字を示しており，その行がどのノードに対して適応されるルールなのかを示している．

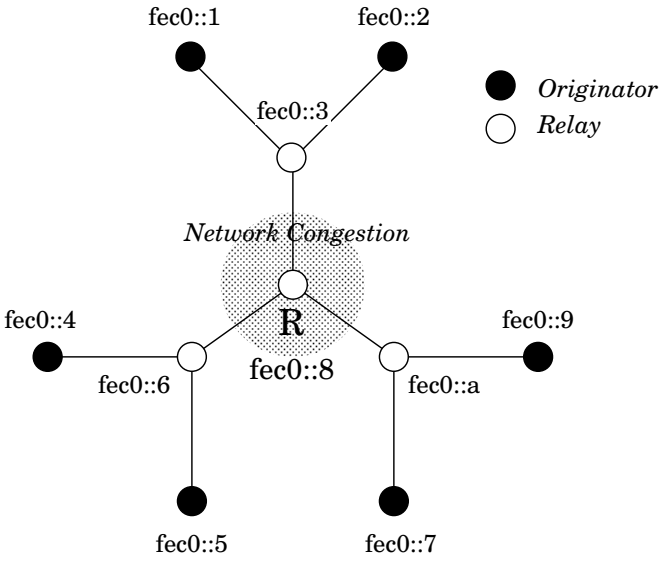

図 8.2: エミュレートするネットワークトポロジ

8.3 実験内容

実験では図 8.2に示した 6 台の送信ノードがフラッディングを行う．ここで重要となるのは，評価項目を評価可能なデータフローを，送信ノードがフラッディングすることである．まず，ネットワークトラフィックの抑制，優先度や Target Thinning Level に応じた削減処理を考慮すると，これらの処理を利用するためにはデータサイズの削減可能なデータが送信される必要がある．さらに，削減処理としてパケットサンプリング処理またはデータ間引き処理を選択する必要がある．本評価ではこれらの処理のうちデータ間引き処理を利用する．これは，パケットサンプリング処理が優先度のみを利用するのに対して，データ間引き処理が優先度，TargetThinning Level の 2 つを利用するため，データ間引き処理の動作を確認することでパケットサンプリング処理の動作も確認出来るからである．以上の理由により，評価実験では各ノードがデータ間引き可能なデータフローに優先度を設定した後，送信する．

次に実験内容を説明する．実験は計 4 回行った．はじめに基礎実験として，AFM を利用しない状態で，送信ノードがデータをフラッディングする．それ以降の実験では，AFM を利用して優先度のみを変更してデータを送信する．次に，実験に関連する事項を列挙する．

- データフローサイズ
  データフローサイズは各実験共に同じである．アドホックネットワークでは様々な種類のトラフィックが想定されるが，評価実験では各ノードが利用帯域の均一なデータフローを送信する．この理由は，本評価実験の目的が，AFM によるネットワークトラフィックの抑制処理を示すことであり，トラフィック種類の違いによる挙動変化は考慮しないためである．本評価実験では，利用帯域の均一なデータフローを送信するためのアルゴリズムを以下の疑似コードに示す．このコードでは，usleep を利用しているため，各ノードの OS の違いにより性能が異なるが，ほぼ一定の通信帯域を利用したデータ送信が可能である．

- データ間引き処理

```
ebtables -X
ebtables -P FORWARD DROP
ebtables -F
ebtables -A FORWARD -o eth0 -s 00:11:24:d0:d1:32 -j ACCEPT #3->1
ebtables -A FORWARD -o eth1 -s 00:11:24:d0:d1:32 -j ACCEPT #3->2
ebtables -A FORWARD -o eth2 -s 00:14:51:17:cf:48 -j ACCEPT #1->3
ebtables -A FORWARD -o eth2 -s 00:14:51:17:d0:a4 -j ACCEPT #2->3
ebtables -A FORWARD -o eth2 -s 00:01:80:5c:0a:cf -j ACCEPT #8->3
ebtables -A FORWARD -o eth3 -s 00:09:6b:a0:a6:71 -j ACCEPT #6->4
ebtables -A FORWARD -o eth4 -s 00:09:6b:a0:a6:71 -j ACCEPT #6->5
ebtables -A FORWARD -o eth5 -s 00:01:80:5c:0a:cf -j ACCEPT #8->6
ebtables -A FORWARD -o eth5 -s 00:11:24:d0:cf:90 -j ACCEPT #4->6
ebtables -A FORWARD -o eth5 -s 00:0d:60:77:fc:39 -j ACCEPT #5->6
ebtables -A FORWARD -o eth6 -s 00:0b:5d:2c:27:a9 -j ACCEPT #a->7
ebtables -A FORWARD -o eth7 -s 00:11:24:d0:d1:32 -j ACCEPT #3->8
ebtables -A FORWARD -o eth7 -s 00:09:6b:a0:a6:71 -j ACCEPT #6->8
ebtables -A FORWARD -o eth7 -s 00:0b:5d:2c:27:a9 -j ACCEPT #a->8
ebtables -A FORWARD -o eth8 -s 00:0b:5d:2c:27:a9 -j ACCEPT #a->9
ebtables -A FORWARD -o eth9 -s 00:01:80:5c:0a:cf -j ACCEPT #8->a
ebtables -A FORWARD -o eth9 -s 00:0d:60:78:c6:b3 -j ACCEPT #7->a
ebtables -A FORWARD -o eth9 -s 00:0d:60:7f:22:60 -j ACCEPT #9->a
```

図 8.3: ebtables によるフォワーディングルールの設定

```
while(1){
  usleep(20000);
  send_packet(1000byte);
}
```

図 8.4: 各ノードが利用する送信アルゴリズム

利用するデータ間引き処理も各実験共に同じである．データ間引きは処理時間を必要とする処理である．本評価実験では，AFM の評価を目的とするため，データ間引き処理に利用するアルゴリズムではパケットサンプリングを利用する，処理時間の少ない方法を利用する．本評価実験で利用するアルゴリズムでは，まず各パケットのデータヘッダを参照することにより，そのデータのシーケンス番号を取得する．その後，AFM ヘッダの Thinning Level を利用することにより 1/2，1/3，1/4，1/5 と段階的にサンプリングレートを上げる．これらの処理によりデータフローサイズを削減する．

- Traffic Threshold

Traffic Threshold は実験 2-4 において利用される．Traffic Threshold は実験 2-4 共に同じ値を設定した．設定した値は，625000byte(5Mbps) である．ネットワークトラフィックはこの値を閾値として抑制されることが予想される．

- Polling Time
  Polling Time も実験 2-4 で利用され，共に同じ値が設定される．Polling Time には 10 秒を設定した．これにより，ネットワーク状態 STALE を検知するまでに 10 秒程度かかることが予想される．

- 優先度
  優先度に関しては実験 2-4 で利用され，互いに異なる値が設定される．実験 2-4 において各ノードに設定した優先度は表に示す通りである．

表 8.2: 実験 1-3 において各データフローに設定される優先度

	fec0::1	fec0::2	fec0::4	fec0::5	fec0::7	fec0::9
基礎実験	-	-	-	-	-	-
実験 1	0	0	0	0	0	0
実験 2	1	2	1	2	2	1
実験 3	1	2	2	3	3	3

以上，実験 1-4 において利用する値を設定した．これらの条件の下，実験では図 8.2におけるノード R(fec0::8) が送受信するデータ総量の変化，送信する各データフローのデータ総量の変化に注目し，各実験共に 120 秒間行った．

8.4 実験結果

本節では評価結果を示す．まず基礎実験の結果を示した後，各実験におけるノード R の送受信データ総量/秒の比較，データフロー別平均データ総量/秒の変化率の比較を行う．

8.4.1 基礎実験

基礎実験の結果を示す．ノード R の受信データ総量/秒のプロット結果を図 8.5にノード R の送受信するデータ総量/秒の平均及び標準偏差，95%信頼区域，99%信頼区域を表 8.3に示す．図 8.5に示す X 軸は時間 (秒)，縦軸は受信データ総量/秒 (KB) を示す．

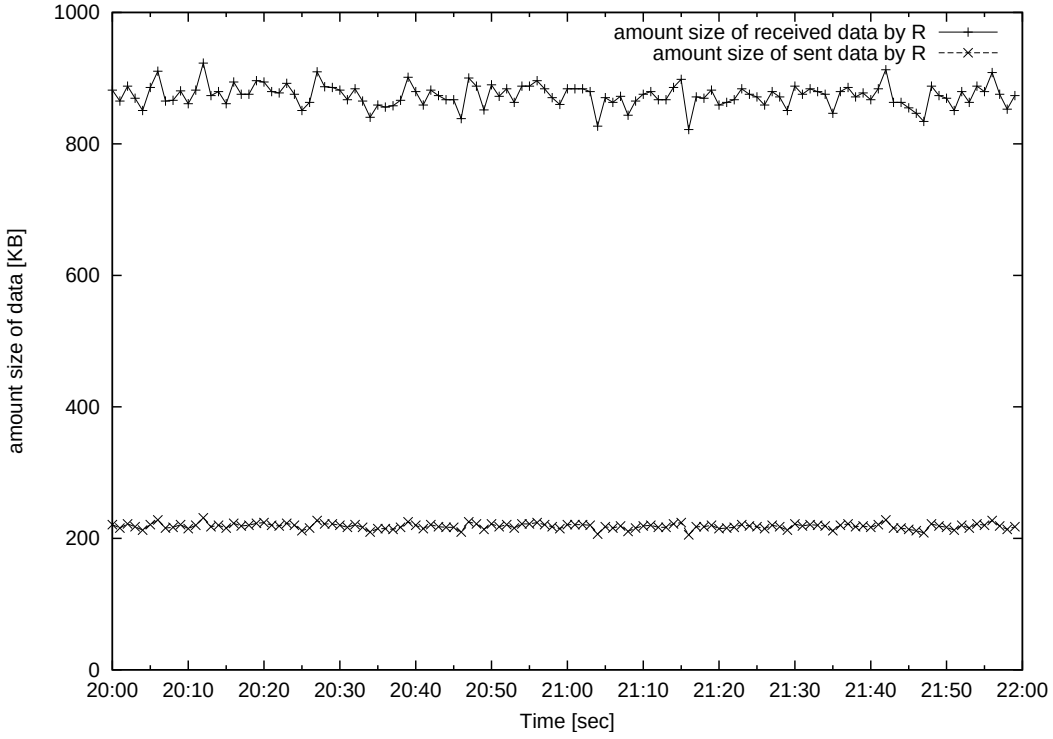

図 8.5: R の受信データ総量/秒 (基礎実験)

表 8.3: データ総量/秒の解析結果 (基礎実験)

	平均 (KB)	標準偏差 (KB)	95%信頼区域 (KB)	99%信頼区域 (KB)
基礎実験 (受信)	874	17	871∼877	870∼878
基礎実験 (送信)	217	20	213∼220	211∼221

基礎実験ではノード R の送受信するデータ総量/秒の平均は約 864KB，約 217KB であった．式 3.1，3.2に示すように中継ノード数に応じて，発生するネットワークトラフィックが増大していることがわかる．また，送受信するデータ総量/秒の標準偏差は受信 17KB，送信 20KB であった．

8.4.2 各実験における受信データ総量/秒の比較

各実験における受信データ総量/秒を比較した．各実験におけるノード R の受信データ総量/秒のプロット結果を図 8.6に，各実験における受信データ総量/ 秒の解析結果を表 8.4に示す．図 8.6に描かれている緑の線は Traffic Threshold である．

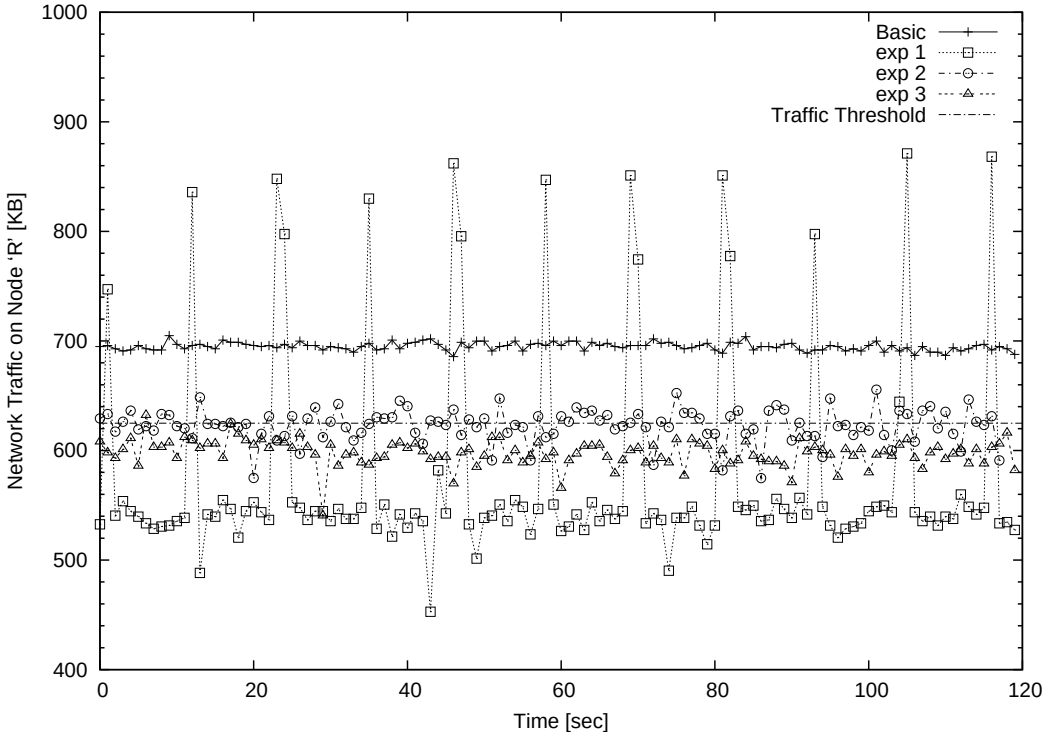

図 8.6: 各実験における受信データ総量/秒の比較

表 8.4: 各実験における受信データ総量/秒の解析結果

	平均 (KB)	標準偏差 (KB)	95%信頼区域 (KB)	99%信頼区域 (KB)
基礎実験	867	17	871∼877	870∼878
実験 1	584	112	563∼604	557∼610
実験 2	638	15	635∼640	634∼642
実験 3	612	12	610∼614	609∼614

図 8.6及び表 8.4に示す受信データ総量/秒の平均値から，AFM を利用することにより受信データ総量/秒が Traffic Threshold 付近まで抑制されることがわかる．次に基礎実験と実験 1-3 の標準偏差について考察する．実験 1 における受信データ総量/秒と基礎実験の標準偏差を比較すると，実験 1 の方が高い．これは，表 8.2に示しすように実験 1 において送信される各データフローの優先度が全て一定であるためである．この場合，図 8.6 に示す実験 1 のように，Polling Time 経過後に Target Thinning Level が下げられることにより，全てのデータフローの間引き率が低くなるためである．一方で実験 2，実験 3 では基礎実験よりも低い標準偏差が得られた．

これは実験 2，実験 3 において各データフローに設定される優先度にばらつきがあるためである．以上の実験結果により AFM を利用することでフラッディング利用時に発生するネットワークトラフィックが抑制されることがわかる．

8.4.3 各実験におけるデータフロー別平均送受信データ総量/秒の変化率の比較

各実験におけるノード R のデータフロー別平均送受信データ総量/秒の変化率を表 8.6に示す．

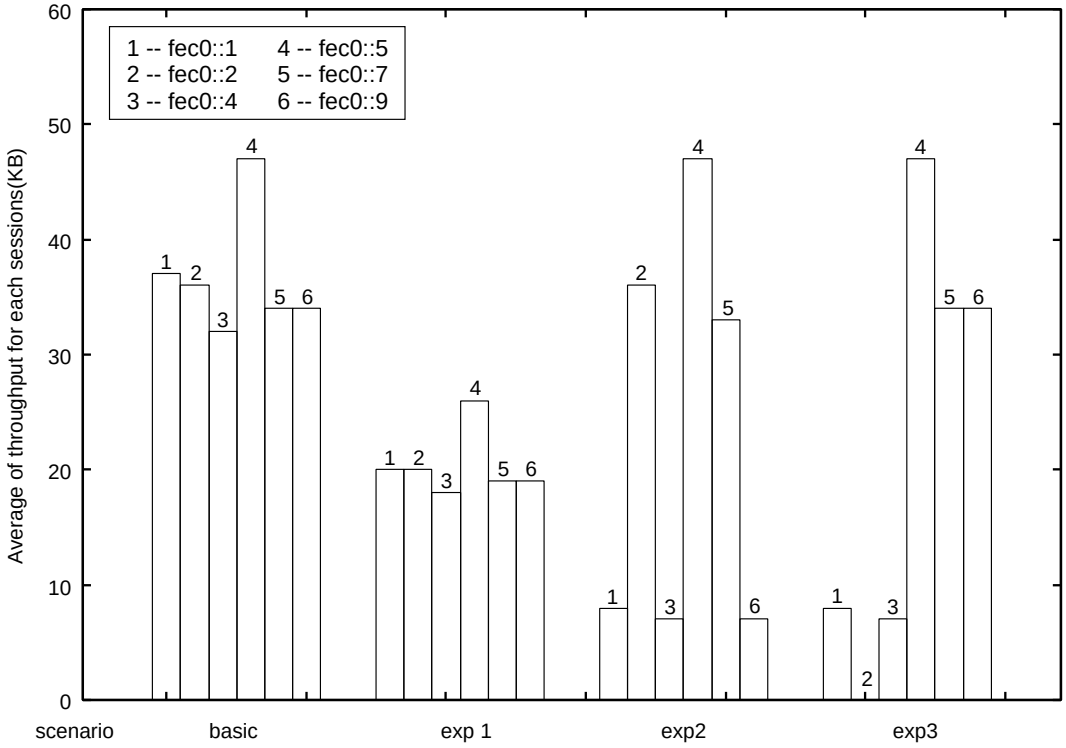

表 8.5: 各実験におけるデータフロー別平均送受信データ総量/秒

表 8.6: 各実験におけるデータフロー別平均送受信データ総量/秒の変化率

	fec0::1	fec0::2	fec0::4	fec0::5	fec0::7	fec0::9
基礎実験	100%	100%	100%	100%	100%	100%
実験 1	56%	54%	55%	55%	56%	56%
実験 2	22%	100%	22%	100%	100%	21%
実験 3	22%	0%	22%	100%	100%	100%

各数値はデータフロー毎にノード R の送信データ総量を受信データ総量で割ることで求めた．図 8.5及び表 8.6から，表 8.2に示す優先度に対応してデータフローが間引かれた後，再送されていることがわかる．これは，AFM が各データフローの優先度及び Target Thinning Level に応じて再送制御及び間引き処理をデータフローに適用することを示す．なお，ノード R の送受信データ総量/秒をフロー別にプロットした図を付録 Bに載せる．

8.5 実験のまとめ

以上実験結果を基に考察を行った．実験結果は AAF の有するデータフローサイズ削減処理として次の 2 点が達成されていることを示した．これにより，AAF がネットワーク状態に応じて自律的なフローコントロール及びネットワークトラフィック管理を実現することがわかる．

- ネットワーク状態に応じたネットワークトラフィックの抑制
- 優先度及び Target Thinning Level に応じたデータフロー削減

なお，本実験では本来データ間引き処理を記述する部分にパケットサンプリングのアルゴリズムを記述したため，データ間引きの処理時間に関しては考慮していない．そのため，実際にアプリケーション依存な処理を記述する場合には処理時間を考慮する必要がある．

第9章　結論

本章では，本論文の結論として，本研究の成果，及び今後の研究課題を示す．

9.1　本研究の成果

本研究では，アドホックネットワークにおけるサービスモデルとしてフラッディングを利用した広告配信モデルを提案した．このサービスモデルを実現するためにはフラッディング利用時の問題である，過剰なネットワークトラフィックの発生を回避する必要がある．本研究では，フラッディング利用時に発生するネットワークトラフィックを抑制するために，次の 4 点に注目した．

- 中継ノード数
- データフロー数
- 各データフローのデータサイズ

本研究では，これらの要因それぞれに対する削減手法を示し，これらの手法を統合して実現するアプリケーション基盤として，Adaptive Application Flooding を提案した．AAF は送受信ノードにおいてはアプリケーション基盤，中継ノードにおいてはフローコンタロール機構として動作する．AAF はフローコントロール機構を実現するために以下の 3 つの機能を持つ．

- 中継ノードの選択機能
- ネットワーク状態及び優先度に応じたデータ間引き機能
- ネットワーク状態及び優先度に応じたパケットサンプリング機能

AAF はネットワークトラフィックを監視することにより，ネットワーク状態の変化を検知する．AAF はデータサイズの削減可能なデータが利用されることを条件に，ネットワーク状態の検知及びデータフローの優先度に応じて，自律的にデータの間引き処理やパケットのサンプリング処理を行う．これらの処理により，AAF はネットワーク状態に適応したコンテンツフラッディングを実現する．また，ネットワークトラフィックの抑制機能以外に，AAF はアプリケーション基盤としての機能を持つことで，複数のアプリケーションから同時に利用出来る．

本研究では，AAF に適応するサンプルアプリケーションを利用することにより，AAF の挙動確認及び評価を行った．評価の結果，AAF がネットワーク状態に応じて，自律的なネットワークトラフィックの抑制を実現することが示された．

9.2 今後の研究課題

本節では今後の研究課題を示す．今後の研究課題に関しては，AAF への追加機能の実装，アドホックネットワーク環境における評価，AAF を利用するアプリケーションの構築，本研究のインターネットへの適用の 4 点が挙げられる．それぞれに関して以降説明する．

9.2.1 Adaptive Application Flooding への追加機能の実装

本研究における Adaptive Application Flooding の実装では，ネットワーク状態の検知は，プログラム起動時に設定する閾値及び消費される通信帯域を利用することにより行われている．しかし，消費される通信帯域の変化だけでなく，無線環境の変化を考慮する必要がある．本研究では，無線環境の変化を測る指標として，SNR(Signal-to-Noise Ratio) の必要性を示したが，本実装では SNR は利用していない．そのため，今後，本実装には SNR を利用したネットワーク環境変化の検知機能を実装し，より詳細なネットワーク状態の検知を実現することを目指す．

9.2.2 アドホックネットワーク環境における評価

Adaptive Application Flooding の評価は第 8章に示したように，トポロジをエミュレートすることにより，本実装の挙動及び性能に関して考察を行った．しかし，本実装の評価が，ノード密度が一定であること，ノードが移動しないことを前提に行われたことを考慮すると，ノード密度の増加やノードの移動の変化に対する本実装の挙動及び性能に関して評価する必要がある．今後の研究ではこれらの評価に関して，まずシミュレーションによる評価を行うことで，本実装の規模性や問題点に関して考察する．さらに，規模性が実現されている場合には，実際のネットワークにおいて動作実験を実施することにより，実際のアドホックネットワークにおける挙動及び性能を評価する必要がある．

9.2.3 Adaptive Application Floodnig を利用するアプリケーションの構築

AAF を利用するアプリケーション例として，距離 (ホップ数) を用いてユーザの再生するコンテンツを変化させるフラッディングアプリケーションが挙げられる．このアプリケーションの持つ特徴を図 9.1に示す．

図 9.1では，送信ノードが音声データを利用して推薦曲を広告している状態を示す．図 9.1中で利用される音声または映像データの受信アプリケーションには，ホップ数に応じて再生する音量，表示ウィンドウサイズを調整する機能が備わっていることを想定する．こうしたアプリケーションを構築することにより，送信ノードから遠くなるにつれて音量，ウィンドウサイズが小さくなるなど，実空間で行われるような，コンテンツの表現が可能になる．また，この特徴は，ネットワーク状態やパケットフォーマットに応じてデータサイズやデータ品質を低下させるという Adaptive Application Flooding の特徴と親和性がある．なぜなら，サンプリング処理によるデータ品質の劣化は，音声データのビットレート，映像データの解像度の低下に繋がるが，共に音量，表示サイズが小さくなることにより，受信者に知覚されにくくなるためである．次に再生するコンテンツを変化させるアプリケーションの持つ利点について説明する．

このアプリケーションの利点は，コンテンツの音量やウィンドウサイズの違いにより，ユーザが送信ノードまでの距離を曖昧に知ることが出来ることである．そのため，コンテンツ内容

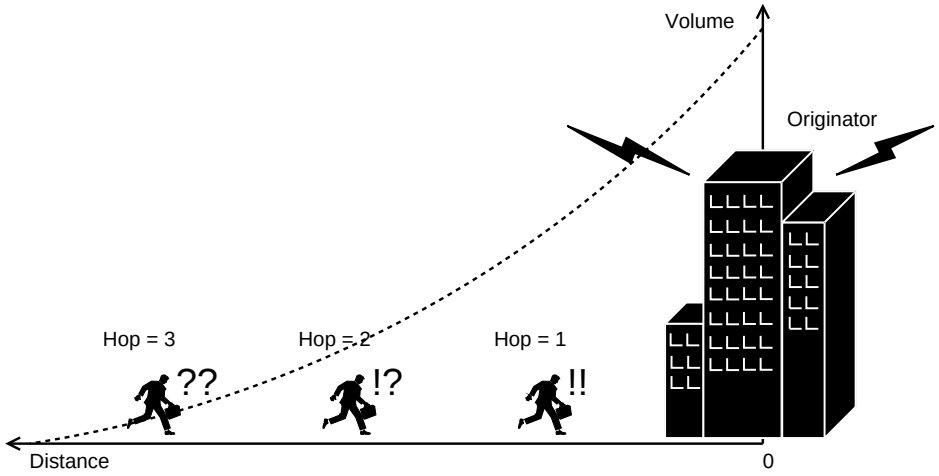

図 9.1: ユーザの再生するコンテンツを変化させるアプリケーション例

に興味を持つ受信者は，より良い再生方法でコンテンツを再生するために，コンテンツ送信場所に近付くといえる．したがって，この特徴は受信者を誘導するという点で広告配信者に対して利益を提供することが可能である．

9.2.4 Autonomous Flow Management のインターネットへの適用

Adaptive Application Flooding を構成するモジュールの一つである Autonomous Flow Management(AFM) モジュールは設計上，選択される Delivery module の種類に関係なく動作可能である．そのため，マルチキャストやその他様々な配信手法との親和性を持つ．また，AFM は中継ノードにおいて利用されることを前提に，アドホックネットワーク以外のネットワークにも適用出来る．例えば，インターネットにおいてもマルチキャスト網が複数重なっている場合には，そのルータに対してネットワークトラフィックが集中することが考慮される．しかし，それらの配信技術全てが AFM を利用する場合，過剰なネットワークトラフィックの発生を防止することが出来る．今後の研究では，インターネットにおけるフローコントロール手法に関してのサーベイや，エンド-エンドモデルと中継ノードモデルに関する比較を行い，それらをまとめた上でインターネットに AFM を適用する．適用の際には，ルータが複数のインタフェース，複数のルーティングエンジンを持つことを考慮する必要がある．

謝辞

本研究を進めるにあたり，御指導を頂きました，慶應義塾常任理事 村井純博士，同大学環境情報学部教授 徳田英幸博士，同学部助教授の中村修博士，楠本博之博士に感謝致します．

本研究を進める上では非常に多くの方から御指導及び御助言を頂きました．特に，心強いサポートをくれた慶應義塾大学専任講師の湧川隆次氏，重近範行氏，杉浦一徳氏には大変感謝しております．多忙な中，筆者の研究の面倒だけでなく，活発な議論の場を提供して下さりありがとうございました．

慶應義塾大学メディア研究科博士過程の石原知洋氏，海崎良氏，三屋光史朗氏はちょっとした質問に対しても真剣に答えて下さりました．また，海外論文を執筆する際にも心強いサポートをして頂きました．慶應義塾大学メディア研究科修士過程の堀場勝広氏，久松剛氏，三島和宏氏，工藤紀篤氏には筆者が研究室に入ってから今まで様々な知識や技術，研究のヒントを与えて頂きました．皆様に追いつけるよう努力していきたいと思っております．良き遊び仲間，クラブ仲間として慶應義塾大学メディア研究科修士の小椋康平氏に感謝致します．筆者の愚痴を面倒がらずに聞いてくれる心の広さに感謝致します．同時期に卒業論文を執筆した水谷正慶氏に感謝致します．心強き仲間として絶大な信頼を寄せております．これからも宜しくお願い致します．いつも心を和ませてくれた後輩として，慶應義塾大学環境情報学部の奥村佑介氏，空閑洋平氏，金井瑛氏に感謝致します．ここに名前を挙げられなかった方々も含め，ここまでこれたのは皆様のおかげです．本当にありがとうございました．

以上を持って謝辞と致します．

参考文献

[1] I. Chakeres, E. Belding-Royer, and C. Perkins. Dynamic MANET On-demand (DYMO) Routing (work in progress, draft-ietf-manet-dymo-03.txt). Internet draft, Internet Engineering Task Force, October 2005.

[2] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc On-Demand Distance Vector (AODV) Routing. Request for Comments (Experimental) 3561, Internet Engineering Task Force, July 2003.

[3] T. Clausen and P. Jacquet. Optimized Link State Routing Protocol OLSR. Request for Comments (Experimental) 3626, Internet Engineering Task Force, October 2003.

[4] T. Clausen and E. Baccelli. OLSRv2 Link Hysteresis (work in progress, draft-clausen-olsrv2-link-hysteresis-00.txt). Internet draft, Internet Engineering Task Force, June 2005.

[5] R. Ogier, F. Templin, and M. Lewis. Topology Dissemination Based on Reverse-Path Forwarding TBRPF. Request for Comments (Experimental) 3684, Internet Engineering Task Force, February 2004.

[6] Y. Yi, S. Lee, W. Su, and M. Gerla. On-Demand Multicast Routing Protocol (ODMRP) for Ad Hoc Networks (work in progress, draft-ietf-manet-odmrp-04.txt). Internet draft, Internet Engineering Task Force, November 2002.

[7] L. Ji and M. S. Corson. Differencial Destination Multicast (DDM) Specification (work in progress, draft-ietf-manet-ddm-00.txt). Internet draft, Internet Engineering Task Force, June 2000.

[8] P. Jacquet, P. Minet, P. Muhlethaler, and N. Rivierre. Increasing reliability in cable free radio LANs: Low level forwarding in HIPERLAN. Wireless Personal Communications, June 1997.

[9] J. Macker and SMF. Design Team. Simplified Multicast Forwarding for MANET (work in progress, draft-ietf-manet-smf-01.txt). Internet draft, Internet Engineering Task Force, October 2005.

[10] V. Paruchuri, A. Durresi, D. Dash, and R. Jain. Optimal flooding protocol for routing in ad-hoc networks. Technical report, Ohio State University, CS Department, 2002.

[11] M. Mauve, J. Widmer, and H. Hartenstein. A survey on position-based routing in mobile ad hoc networks. IEEE Network Magazine, 15(6):30–39., November 2001.

[12] Julio C. Navas and Tomasz Imielinski. GeoCast – geographic addressing and routing. In *Mobile Computing and Networking*, pages 66–76, 1997.

[13] Y. Ko and N. Vaidya. GeoTORA: A protocol for geocasting in mobile ad hoc networks, 2000.

[14] S. McCanne, V. Jacobson, and M. Vettterli. Receiver-driven layered multicast. in Proceedings IEEE Infocom '96, San Francisco, CA, March 1996.

[15] K.Nakauchi, H. Morikawa, and T. Aoyama. A Network-Supported Approach to Layered Multicast. in Proceedings IEEE ICC, Helsinki, Finland, June 2001.

[16] A. Qayyum, L. Viennot, and A. Laouiti. Multipoint relaying: An efficient technique for flooding in mobile wireless networks. INRIA research report RR-3898, March 2000.

[17] A.Qayyum, L. Viennot, and A. Laouiti. Multipoint Relaying for Flooding Broadcast Messages in Mobile Wireless Networks. 35th Annual Hawaii Internationial Conference on System Sciences (HICSS'02)-Volume9, 2002.

[18] C. Adjih, P. Jacquet, and L. Viennot. Computing connected dominated sets with multipoint relays. Technical Report, INRIA, October 2002.

[19] T. Narten, E. Nordmark, and W. Simpson. Neighbor Discovery for IP Version 6 (IPv6). Request for Comments (Standards Track) 2461, Internet Engineering Task Force, December 1998.

[20] J. Postel. Internet Protocol. Request for Comments (Standards Track) 791, Internet Engineering Task Force, September 1981.

[21] S. Deering and R. Hinden. Internet Protocol, Version 6. Request for Comments (Standards Track) 1883, Internet Engineering Task Force, December 1995.

[22] S. Blake, D. Black, M. Carlson E. Davies, Z. Wang, and W. Weiss. An Architecture for Differentiated Services. Request for Comments (Informational) 2475, Internet Enginerring Task Force, December 1998.

[23] K. Nichols, S. Blake, F. Baker, and D. Black. Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers. Request for Comments (Standards Track) 2474, Internet Enginerring Task Force, December 1998.

[24] K. Ramakrishnan, S. Floyd, and D. Black. The Addition of Explicit Congestion Notification (ECN) to IP. Request for Comments (Standards Track) 3168, Internet Enginerring Task Force, September 2001.

[25] N. Spring, D. Wetherall, and D. Ely. Robust Explicit Congestion Notification (ECN) Signaling with Nonces. Request for Comments (Experimental) 3540, Internet Enginerring Task Force, June 2003.

[26] H. Arora and H. Sethu. A Simulation Study of the Feasibility of Differentiated Services Framework for QoS in Mobile Ad Hoc Networks. Proceedings of the Applied Telecommunication Symposium, San Diego, apr 2002.

[27] 大薮 勇輝, 久松 剛, 杉浦 一徳, 中村 修, 村井 純. DSCP(DV Stream Control Proxy) の設計と実装. マルチメディア通信と分散処理研究会, December 2004.

[28] A. Ogawa. *DVTS (Digital Video Transport System) WWW page*, November 2001. URL:http://www.sfc.wide.ad.jp/DVTS/.

[29] J. Macker and SMF. Design Team. Simplified Multicast Forwarding for MANET (work in progress, draft-ietf-manet-smf-00.txt). Internet draft, Internet Engineering Task Force, June 2005.

[30] EBCOT: Embedded Block Coding with Optimized Truncation. ISO/IEC JTC1/SC29/WG1 N1020R, October 1998.

[31] A. Islam and W. A. Perlman. Set partitioned sub-block coding (SPECK). ISO/IEC/JTC1/SC29, WG1 N1188, June 1998.

[32] Extending the MPEG-4 AAC Codec by Perceptual Noise Substitution. 104st Convention of the Audio Engineering Society, preprint 4720.

[33] Transform domain weighted interleave vector quantization (Twin VQ). 101st Convention of the Audio Engineering Society, preprint 4377.

[34] Generic Coding of Moving Pictures and Associated Audio. ISO/IEC International Standard 13818, 1994.

[35] ISO/IEC 15444-3 Motion-JPEG2000 (JPEG2000 Part 3), 2000.

付 録A　階層符号化技術

階層符号化圧縮されたデータはBaseレイヤ，Enhancedレイヤ，2種類のレイヤから構成される．それぞれのレイヤは，一定のデータ品質を持っており，Baseレイヤは最低限の品質を提供する．Enhancedレイヤは，Baseレイヤに対する上位レイヤとして構成され，Enhancedレイヤを重ねるにつれて，データ品質を向上させることが出来る．このデータ品質は，映像や画像の場合は解像度や情報量，音声の場合はビットレートを指す．階層的な構造を持つデータでは，Enhancedレイヤ数を変更することにより，再生時のデータ品質だけでなく，データサイズを変更することが出来る．

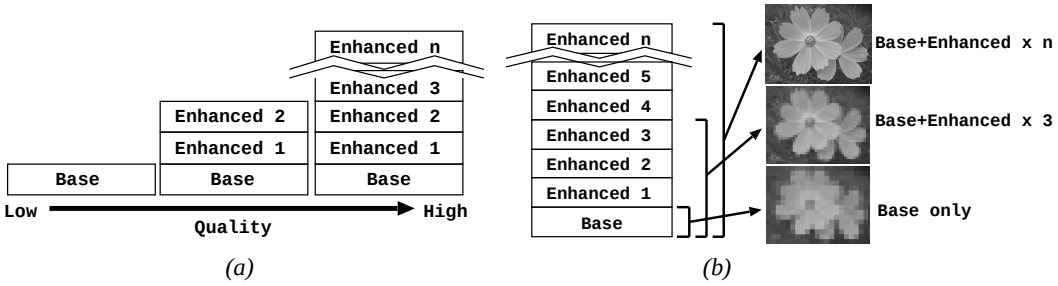

図 A.1: 階層的な構造を持つデータ

階層符号化圧縮を利用するコーデックとしてはJPEG-2000(EBCOT[30])が有名である．JPEG-2000(EBCOT)は画像データを階層符号化圧縮するために利用される．JPEG-2000(EBCOT)では解像度の異なる複数の画像を用意した後，最も品質の低い画像データをBaseレイヤとして符号化する．さらに，Baseレイヤと解像度の高い画像データとの差分を符号化したものをEnhancedレイヤとする．このように，下位レイヤを基に上位レイヤが符号化される．階層符号化エンコーディングを適用したデータは図A.1(a)に示すようにより多いレイヤ数を選択することによりLOWからHIGHにかけて，データ品質が向上する．図A.1(b)は，階層符号化された画像データからEnhancedレイヤ数を減少させることで，解像度が低下していく様子を示している．

階層符号化圧縮を実現するコーデック及びアルゴリズムは画像や音声，映像，それぞれに関して提案されている．画像に関してはJPEG-2000(EBCOT)，QccPack(SPECK[31])，音声に関してはMPEG4(AAC-SSR[32])，MPEG4(TwinVQ[33])，映像に関してはMPEG2(SNR Profile)[34]，MotionJPEG2000[35]などが挙げられる．

付 録B　実験データ

本付録では第 8章における実験結果の補足資料を載せる．補足内容は以下の通りである．

- 各実験におけるノード R の送信データ総量/秒の解析結果
- 実験 1-3 におけるノード R のデータフロー別平均送受信データ総量/秒
- 各実験におけるノード R の送受信データ総量/秒のプロット結果
- 実験 1-3 におけるード R のデータフロー別送受信データ総量/秒のプロット結果

表 B.1: 各実験におけるノード R の送信データ総量/秒解析結果

	平均 (KB)	標準偏差 (KB)	95%信頼区域 (KB)	99%信頼区域 (KB)
基礎実験	217	20	213∼220	212∼221
実験 1	122	35	116∼128	114∼130
実験 2	138	13	135∼140	135∼141
実験 3	130	12	128∼132	127∼133

表 B.2: 実験 1-3 におけるノード R のデータフロー別送受信データ総量/秒

		fec0::1	fec0::2	fec0::4	fec0::5	fec0::7	fec0::9
実験 1	受信 (KB)	37	35	33	47	34	34
	送信 (KB)	20	20	18	26	19	19
実験 2	受信 (KB)	36	36	32	46	33	34
	送信 (KB)	8	36	7	46	33	7
実験 3	受信 (KB)	37	35	32	47	34	34
	送信 (KB)	8	0	7	47	34	34

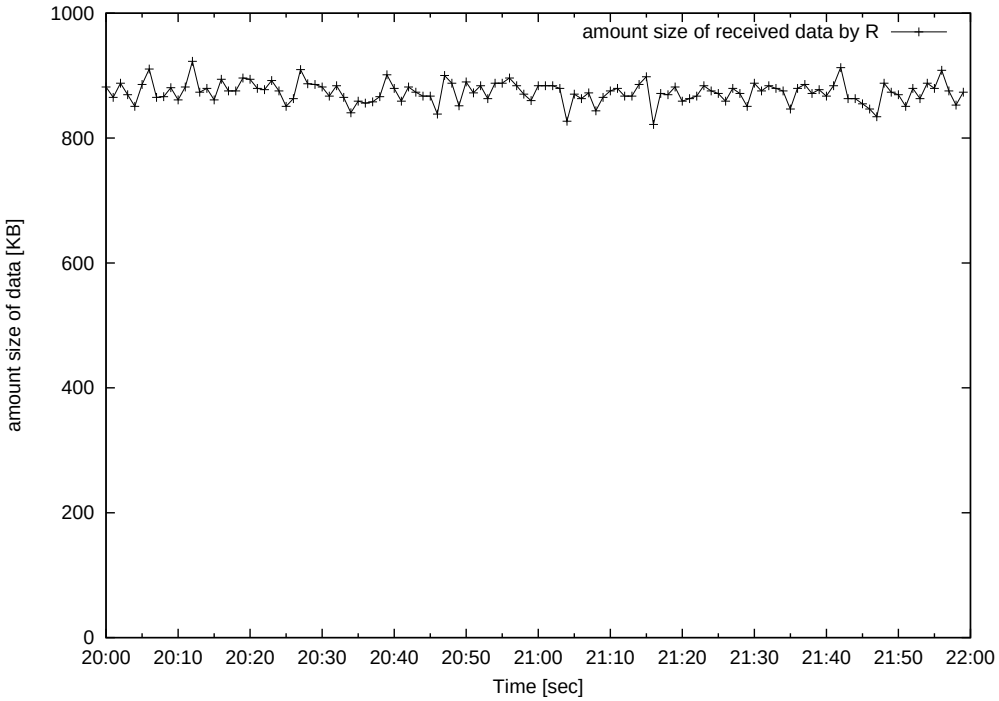

図 B.1: R の受信データ総量/秒 (基礎実験)

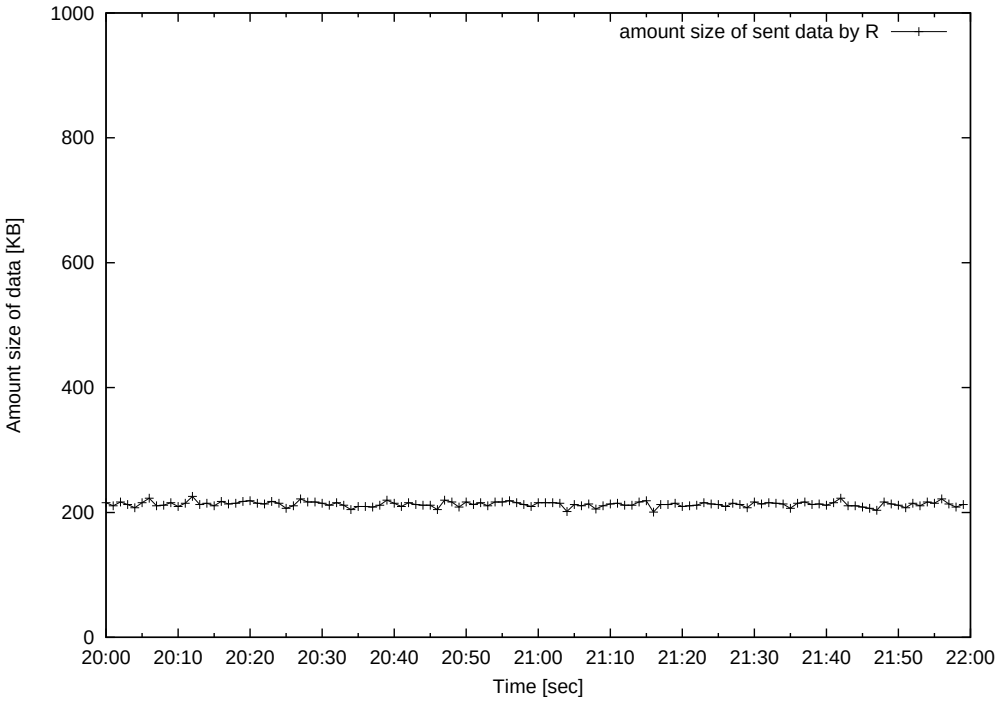

図 B.2: R の送信データ総量/秒 (基礎実験)

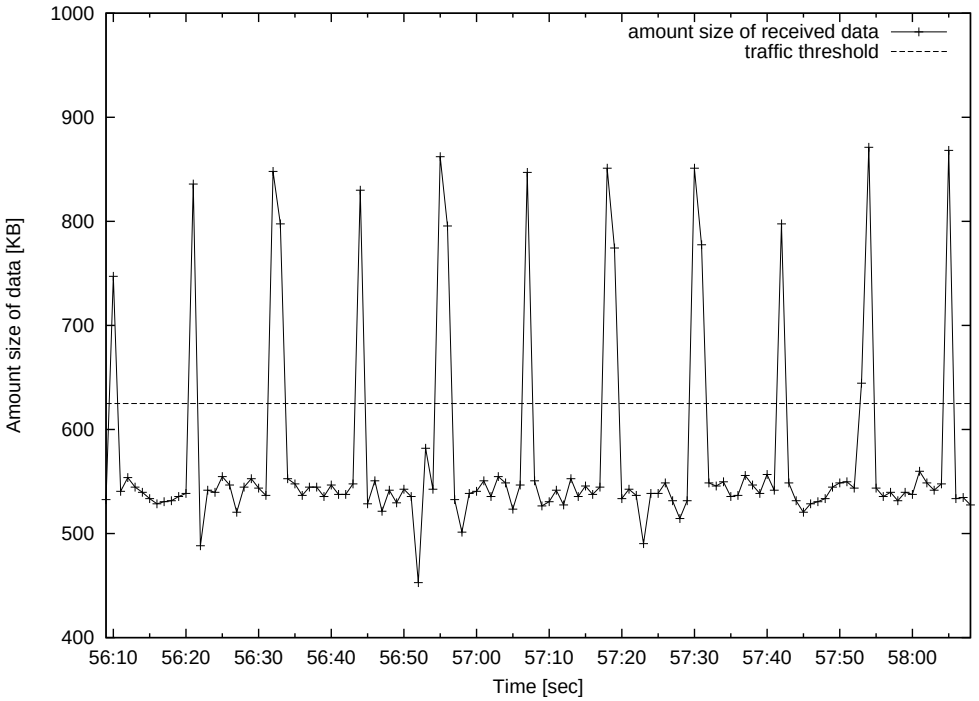

図 B.3: R の受信データ総量/秒 (実験 1)

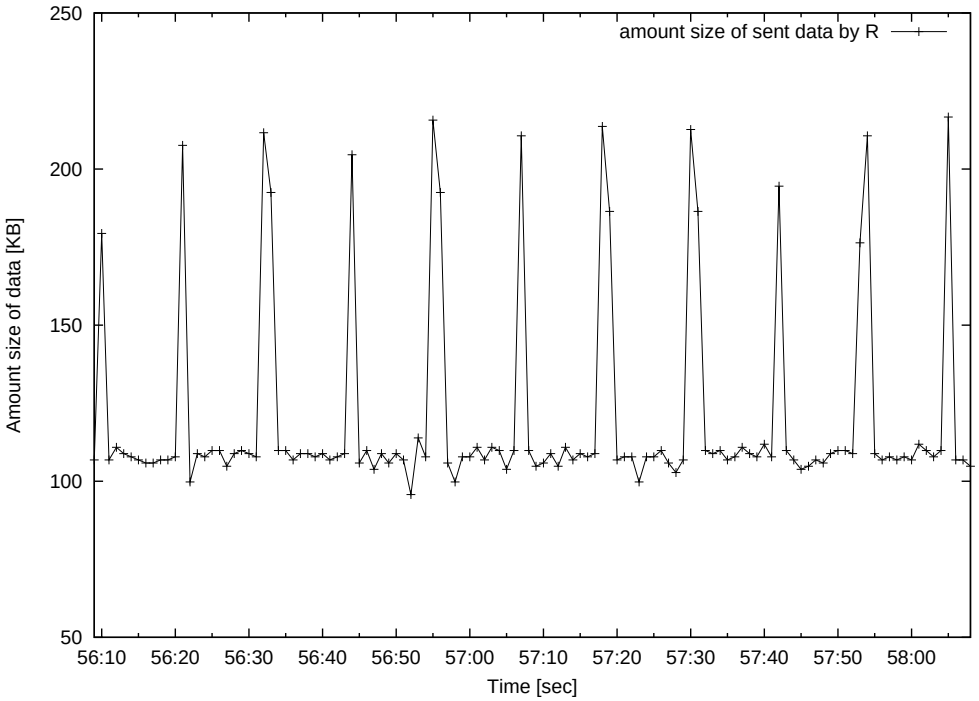

図 B.4: R の送信データ総量/秒 (実験 1)

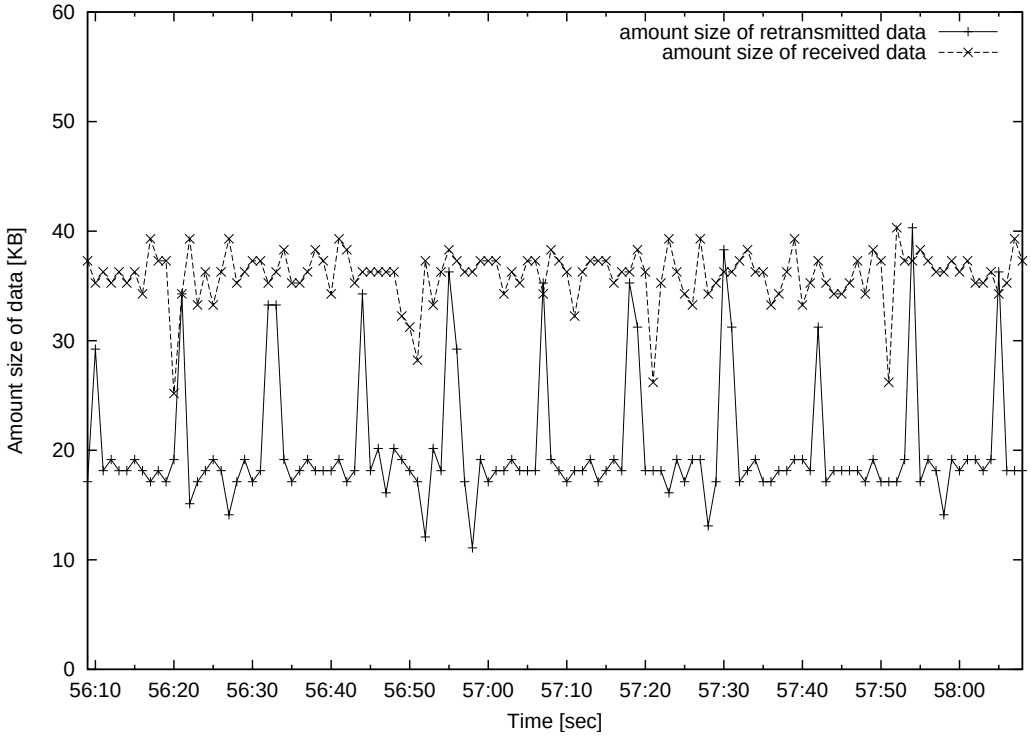

図 B.5: fec0::1 のデータフローに関する R の送受信データ総量/秒の比較 (実験 1)

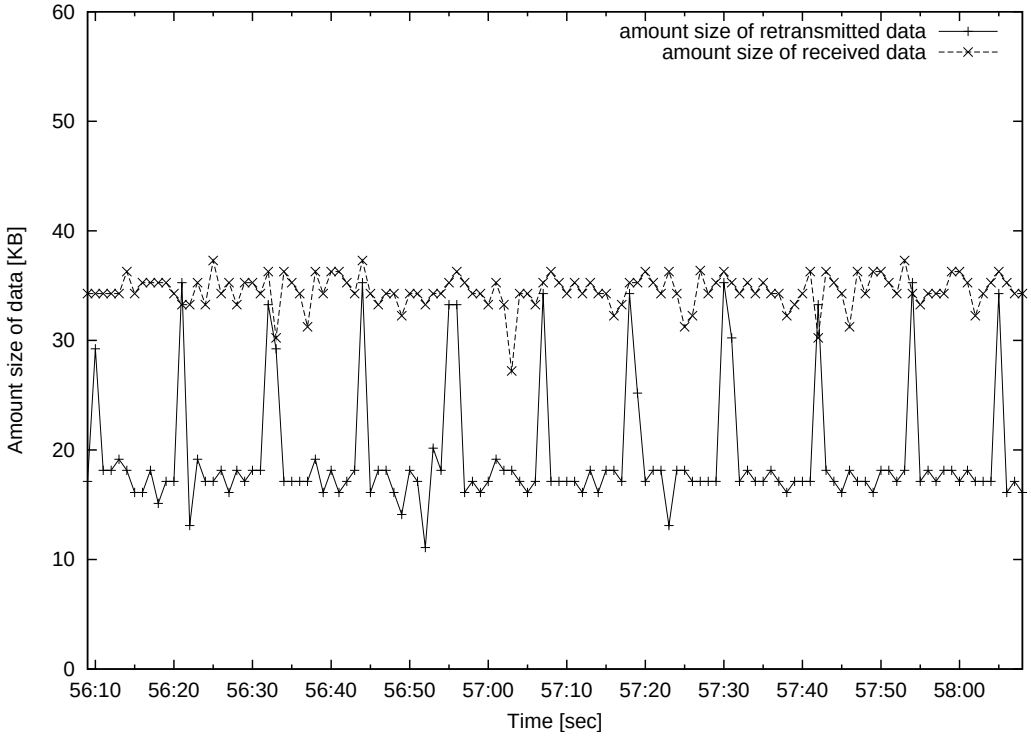

図 B.6: fec0::2 のデータフローに関する R の送受信データ総量/秒の比較 (実験 1)

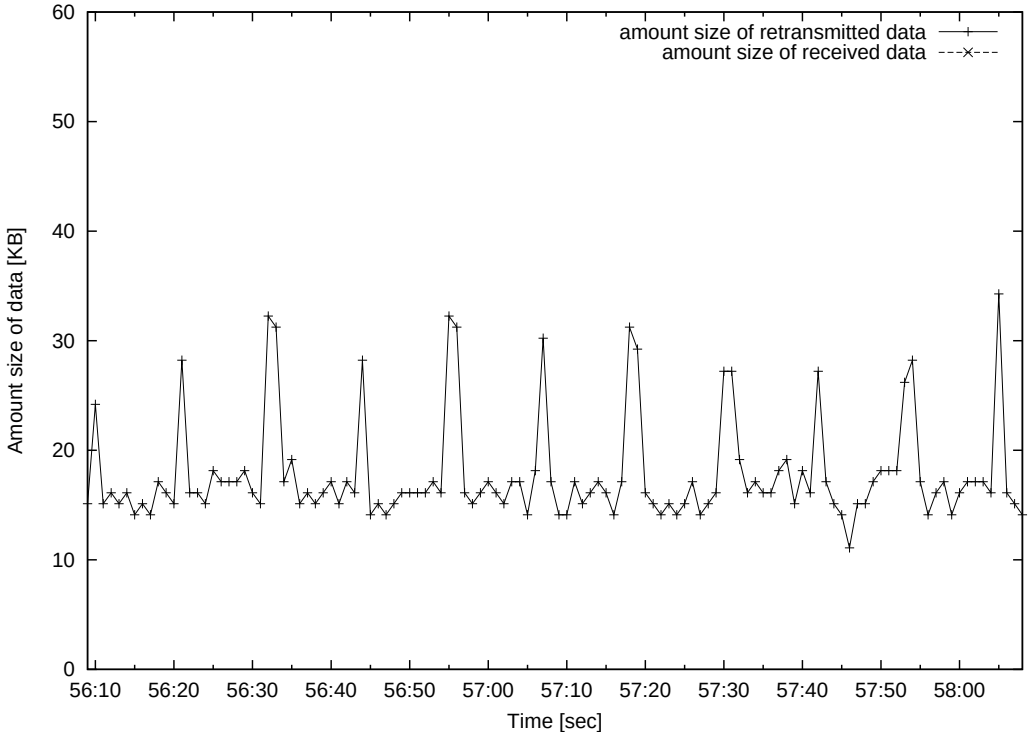

図 B.7: fec0::4 のデータフローに関する R の送受信データ総量/秒の比較 (実験 1)

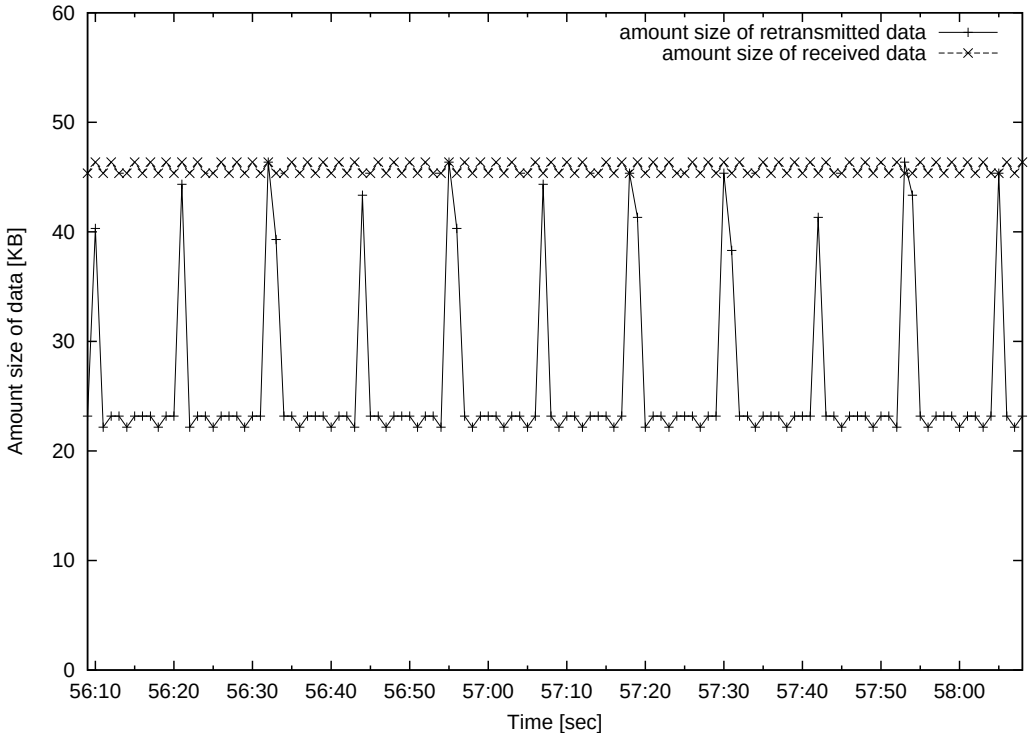

図 B.8: fec0::5 のデータフローに関する R の送受信データ総量/秒の比較 (実験 1)

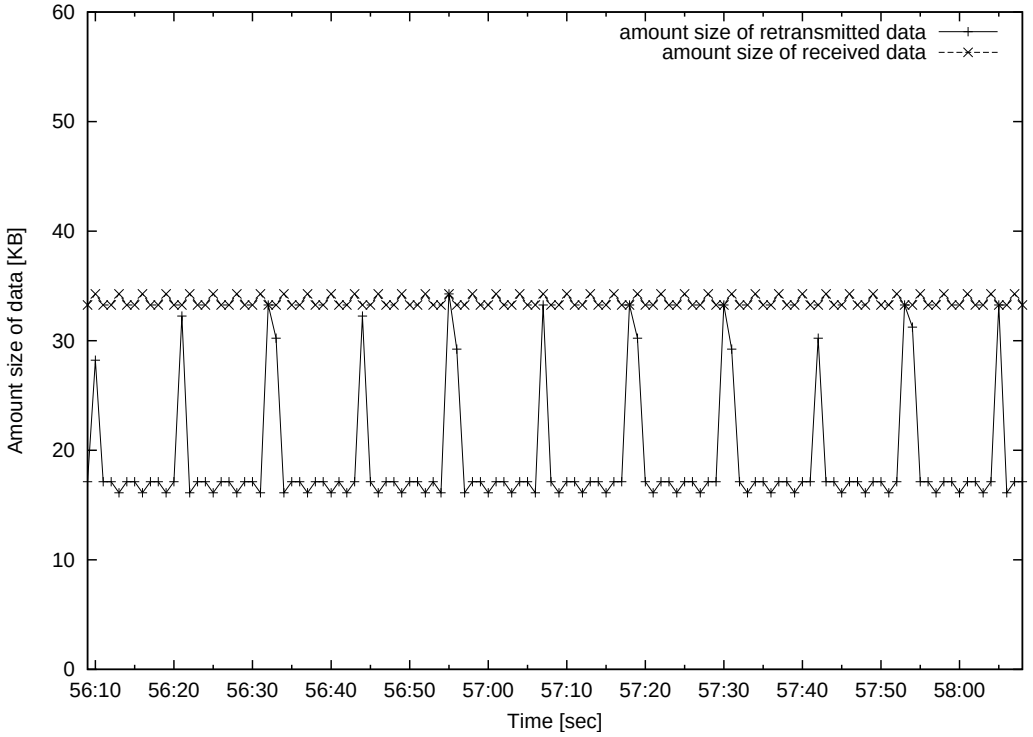

図 B.9: fec0::7 のデータフローに関する R の送受信データ総量/秒の比較 (実験 1)

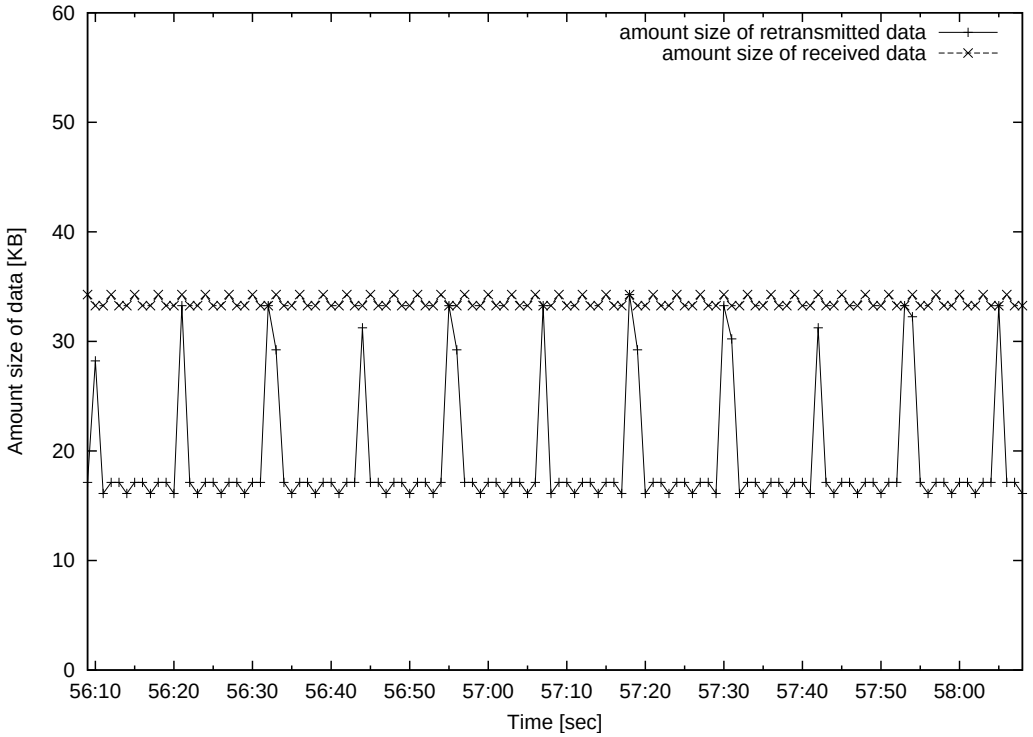

図 B.10: fec0::9 のデータフローに関する R の送受信データ総量/秒の比較 (実験 1)

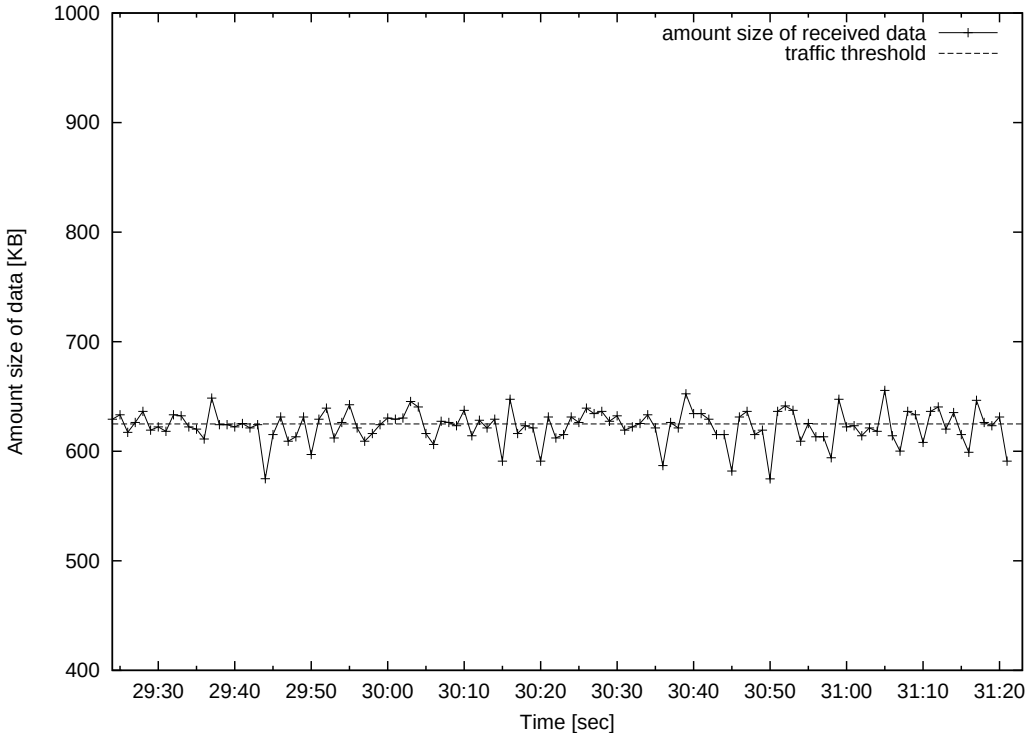

図 B.11: R の受信データ総量/秒 (実験 2)

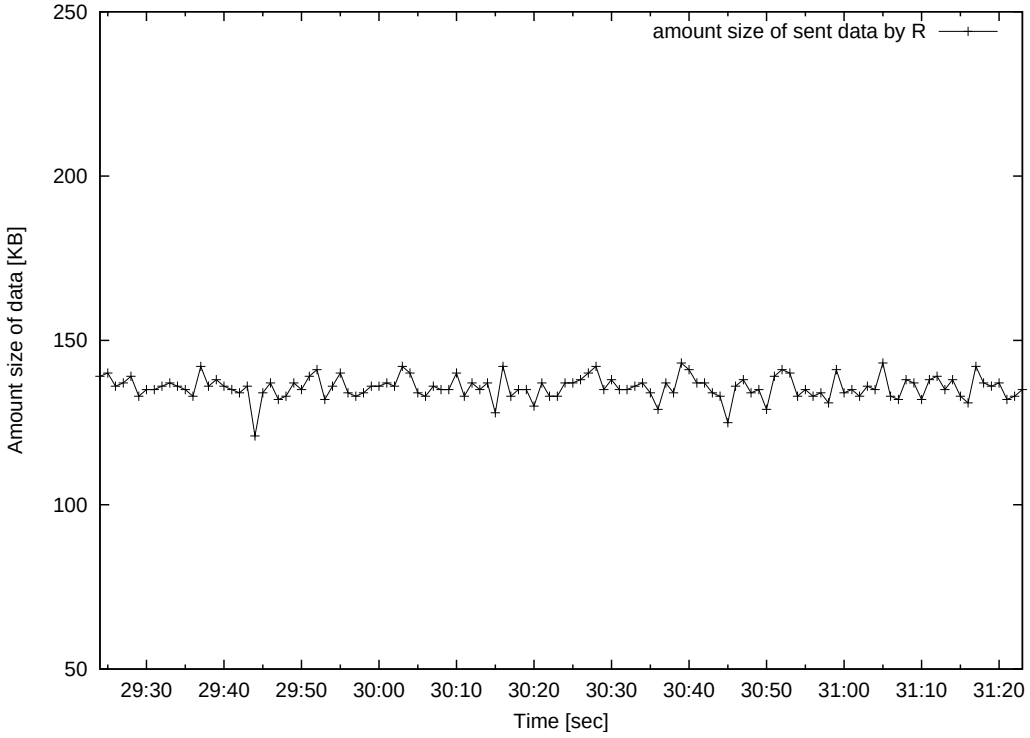

図 B.12: R の送信データ総量/秒 (実験 2)

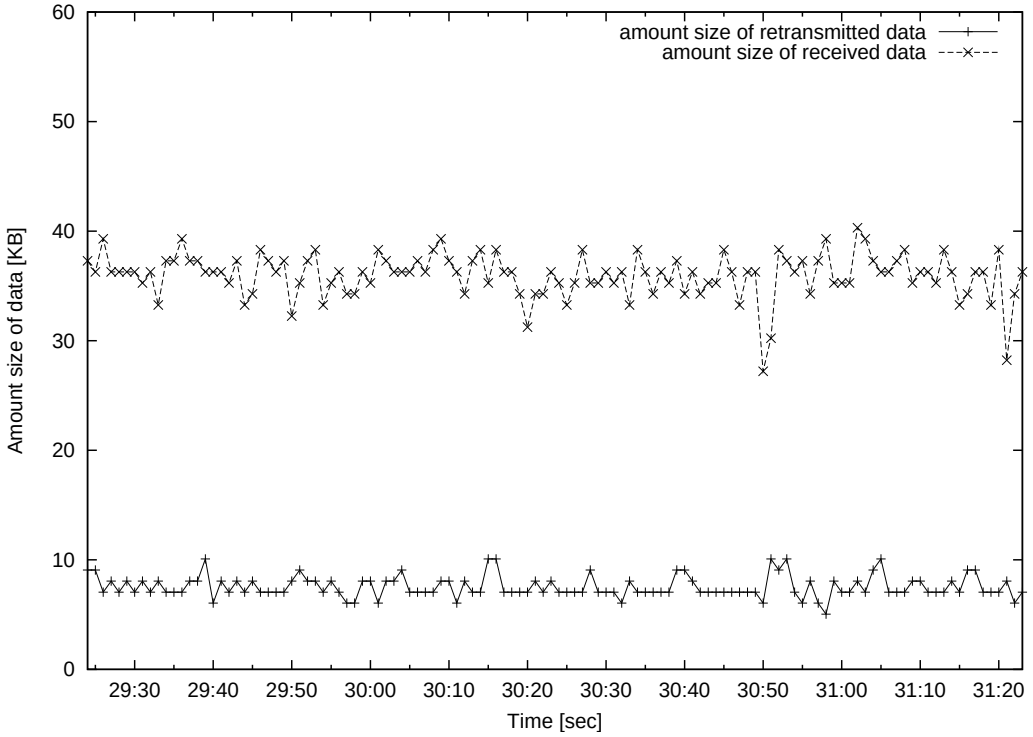

図 B.13: fec0::1 のデータフローに関する R の送受信データ総量/秒の比較 (実験 2)

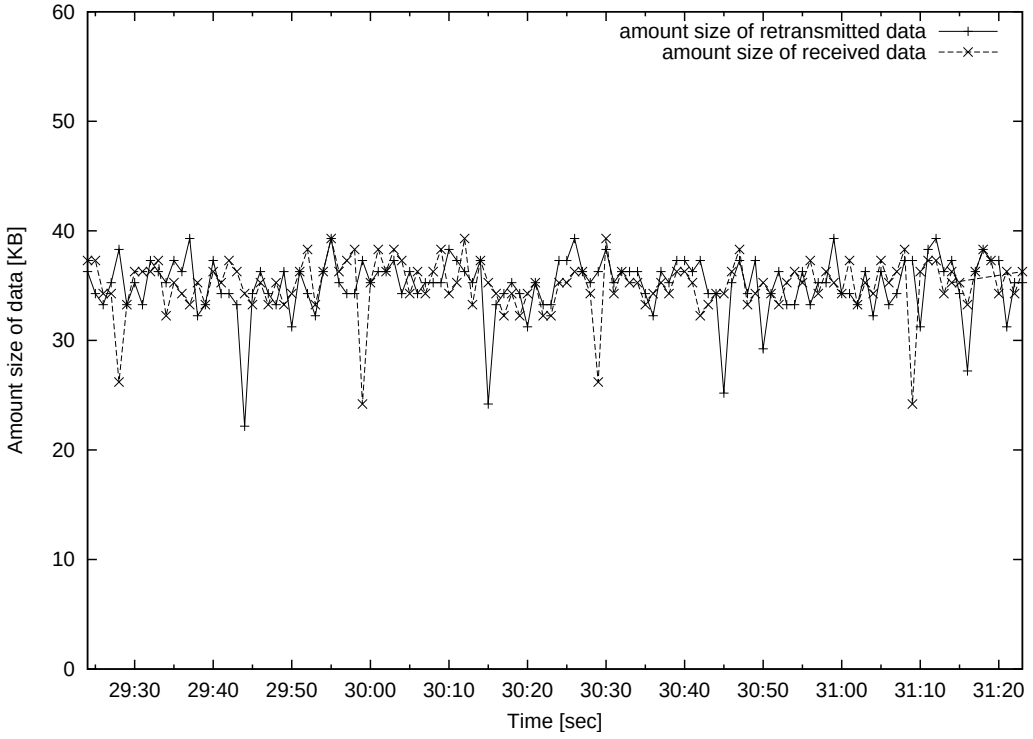

図 B.14: fec0::2 のデータフローに関する R の送受信データ総量/秒の比較 (実験 2)

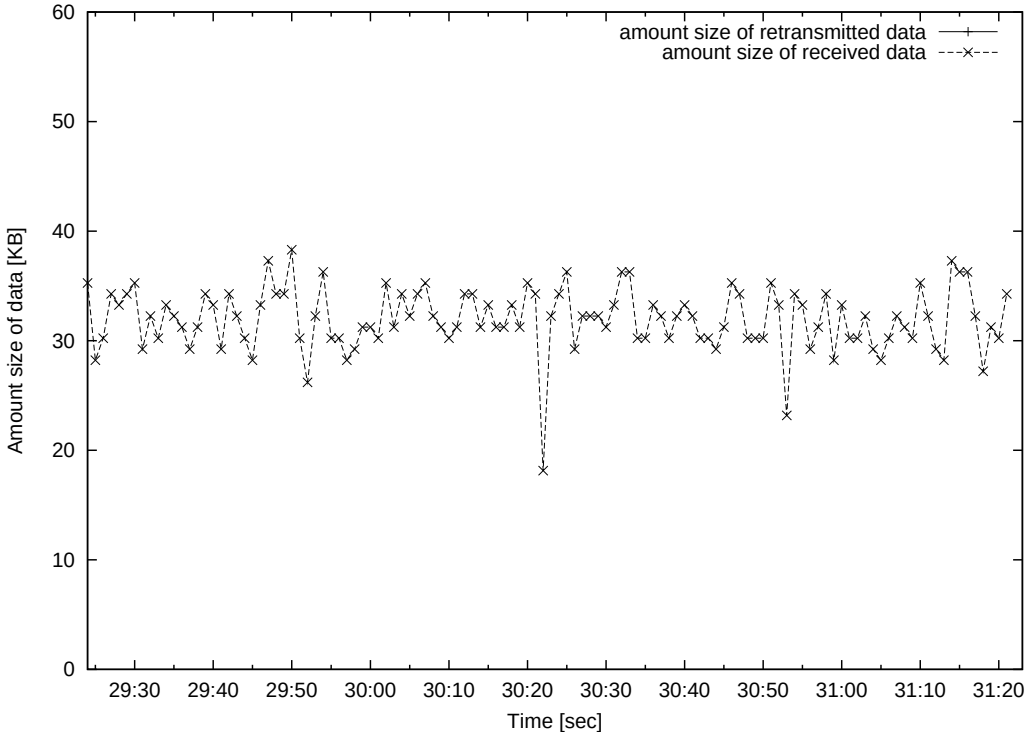

図 B.15: fec0::4 のデータフローに関する R の送受信データ総量/秒の比較 (実験 2)

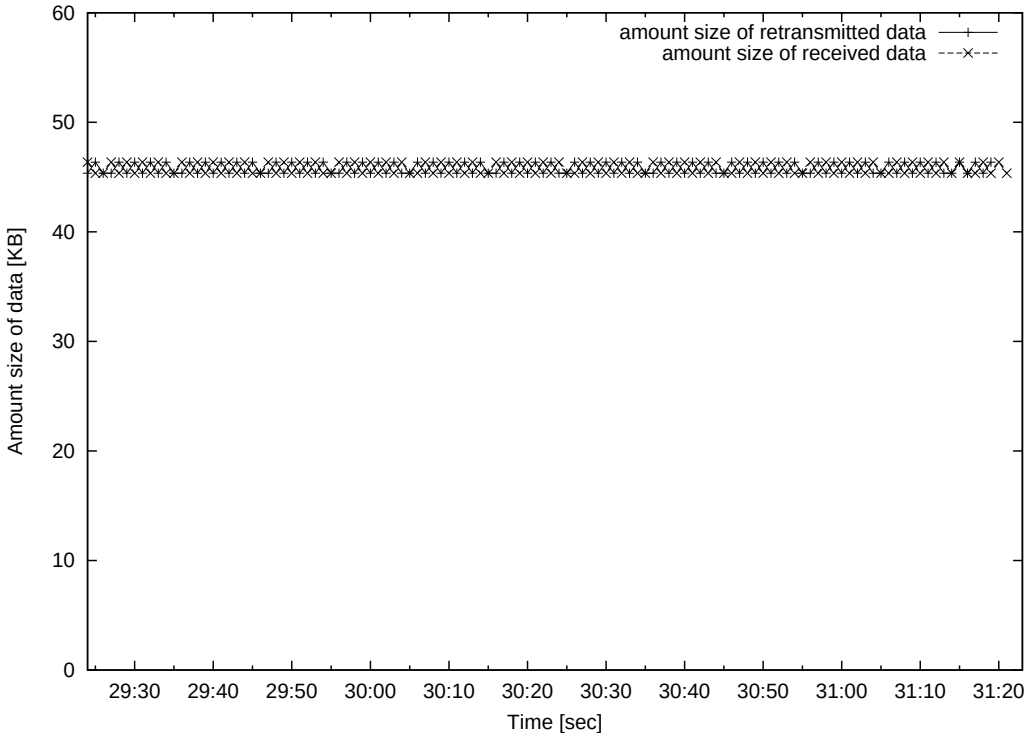

図 B.16: fec0::5 のデータフローに関する R の送受信データ総量/秒の比較 (実験 2)

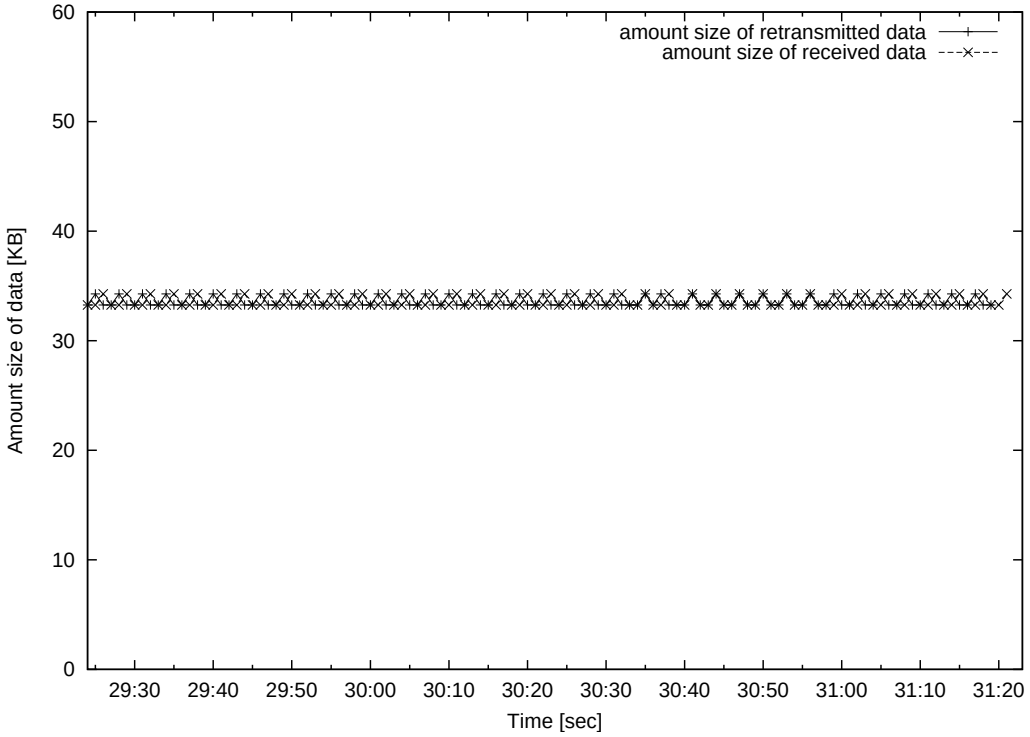

図 B.17: fec0::7 のデータフローに関する R の送受信データ総量/秒の比較 (実験 2)

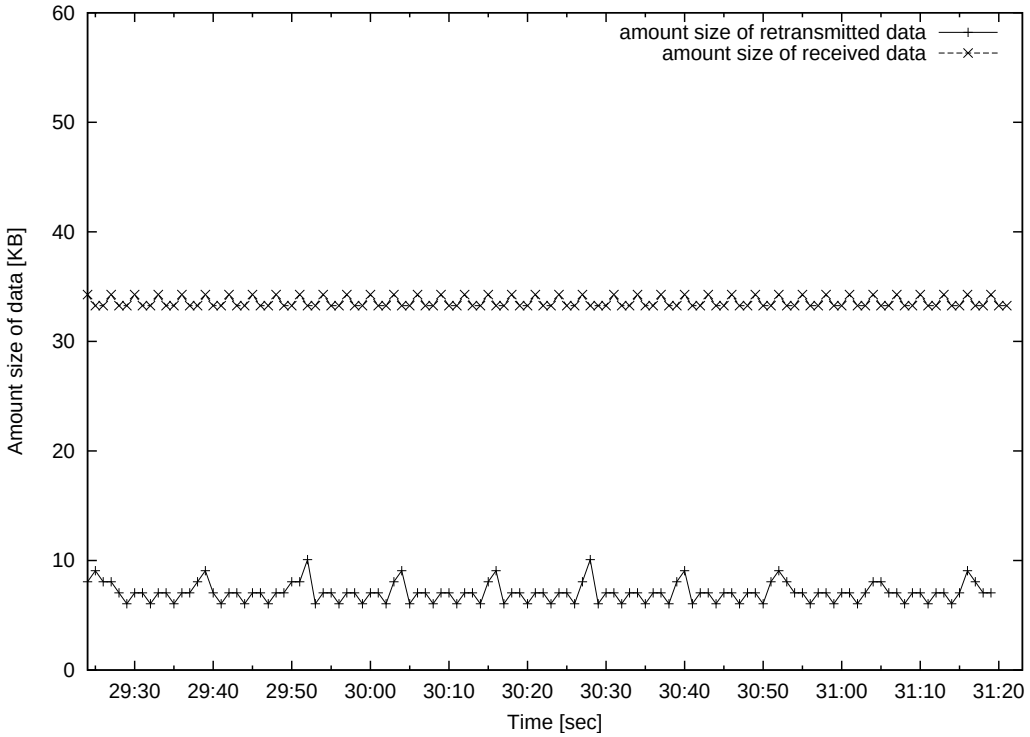

図 B.18: fec0::9 のデータフローに関する R の送受信データ総量/秒の比較 (実験 2)

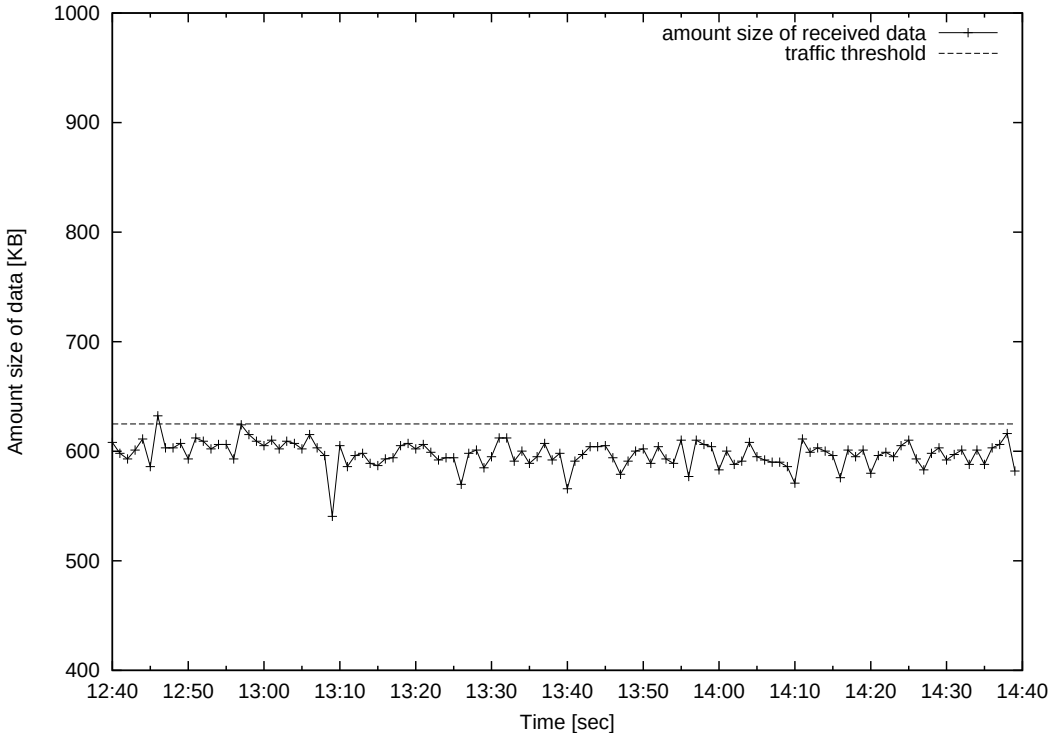

図 B.19: R の受信データ総量/秒 (実験 3)

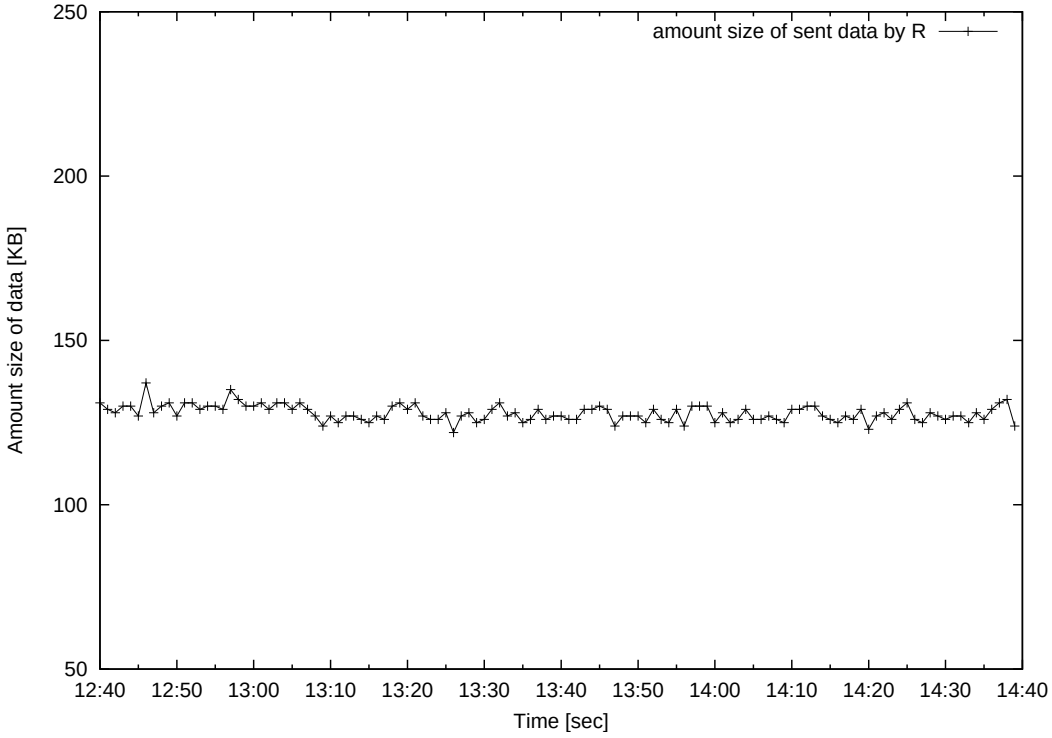

図 B.20: R の送信データ総量/秒 (実験 3)

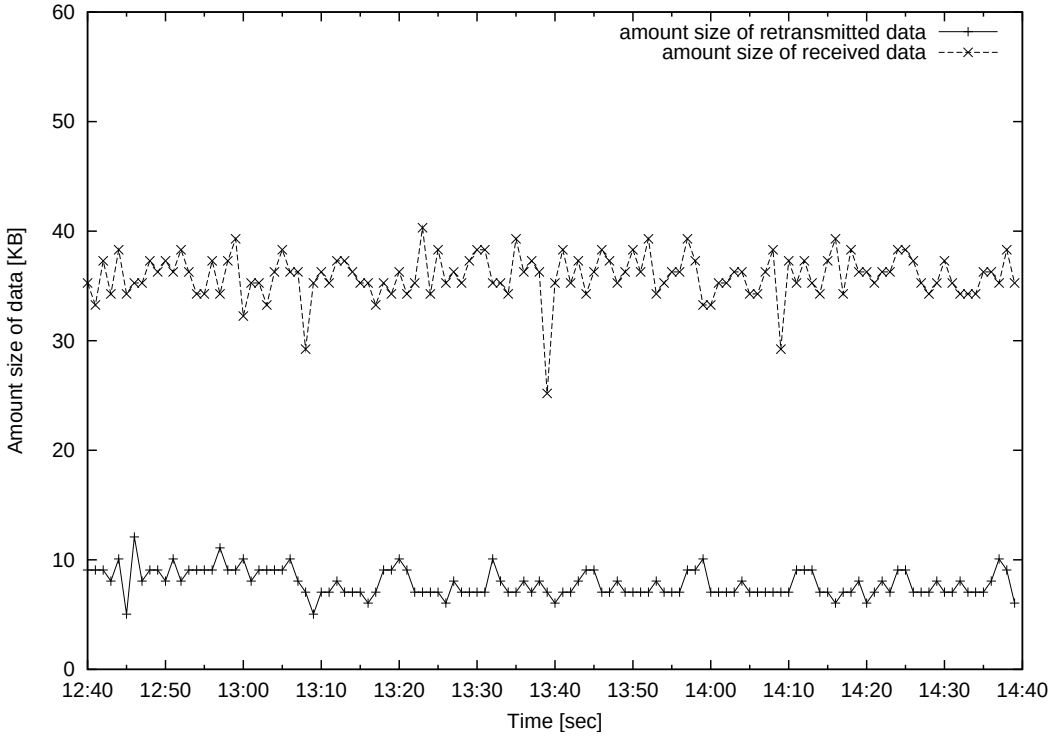

図 B.21: fec0::1 のデータフローに関する R の送受信データ総量/秒の比較 (実験 3)

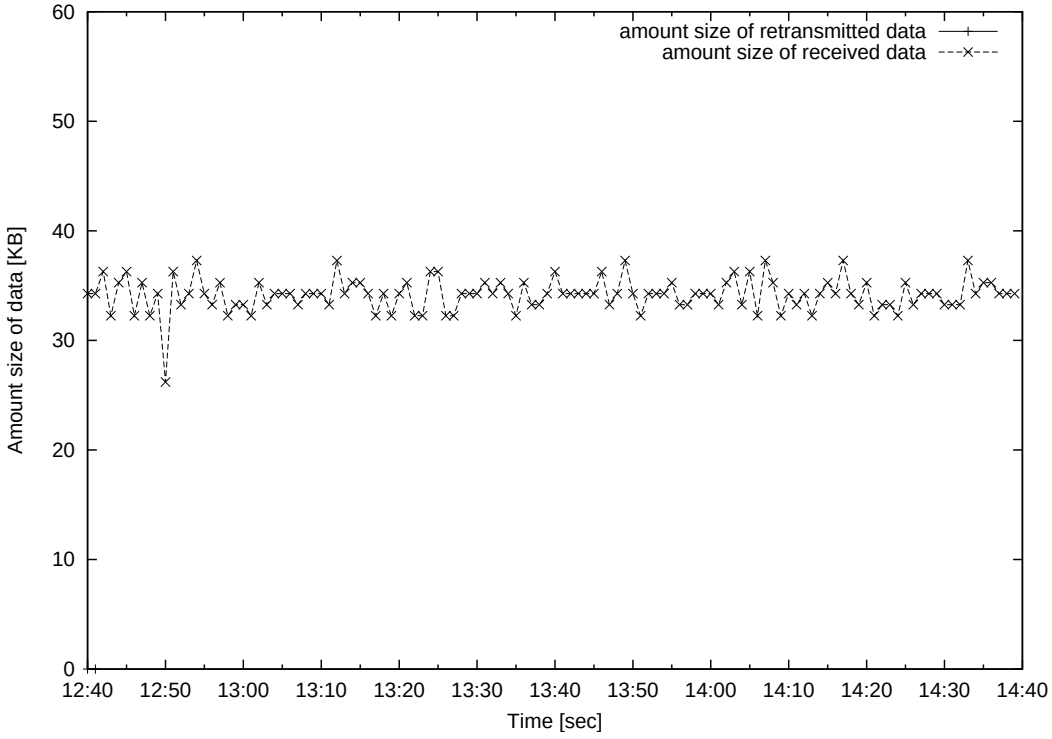

図 B.22: fec0::2 のデータフローに関する R の送受信データ総量/秒の比較 (実験 3)

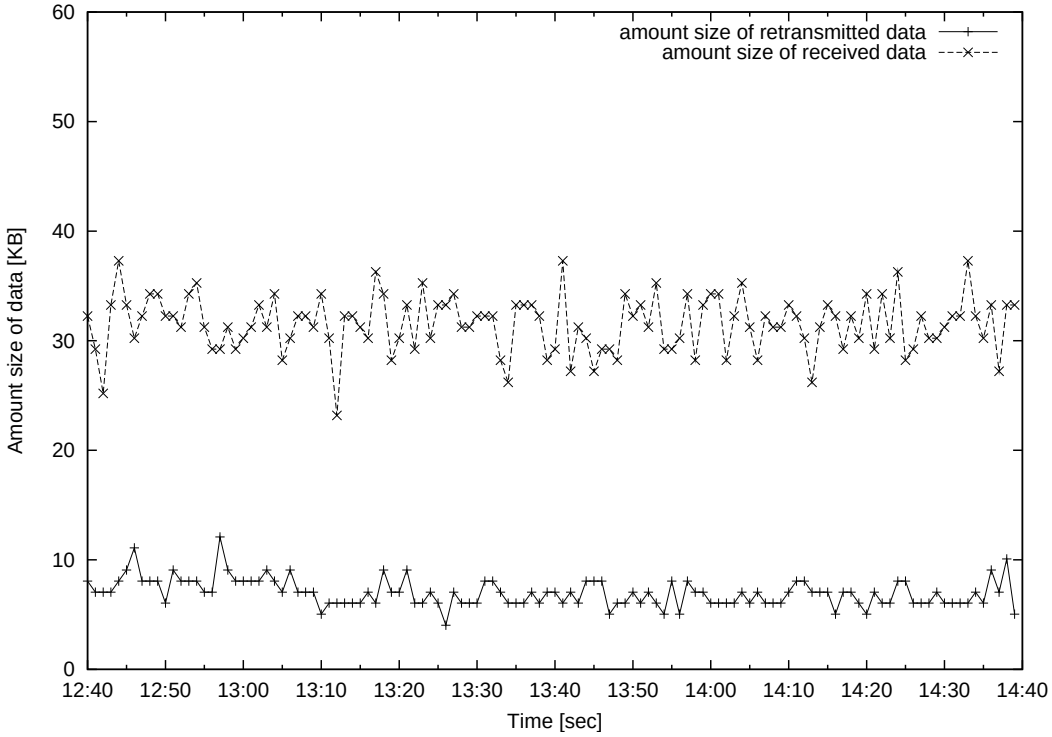

図 B.23: fec0::4 のデータフローに関する R の送受信データ総量/秒の比較 (実験 3)

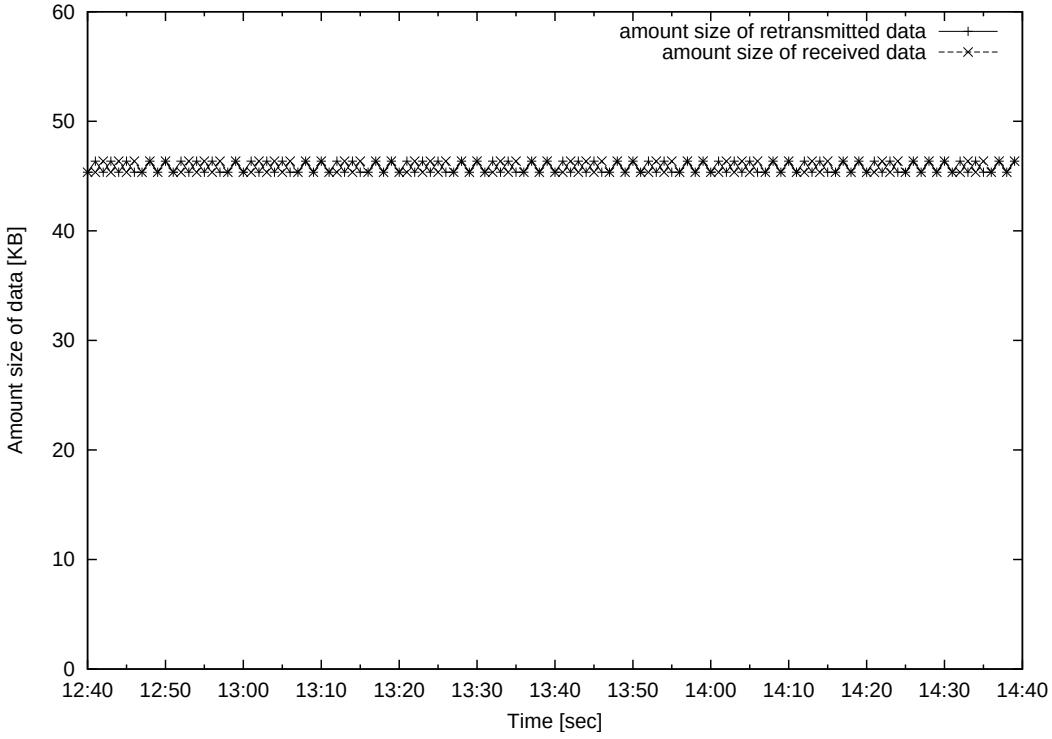

図 B.24: fec0::5 のデータフローに関する R の送受信データ総量/秒の比較 (実験 3)

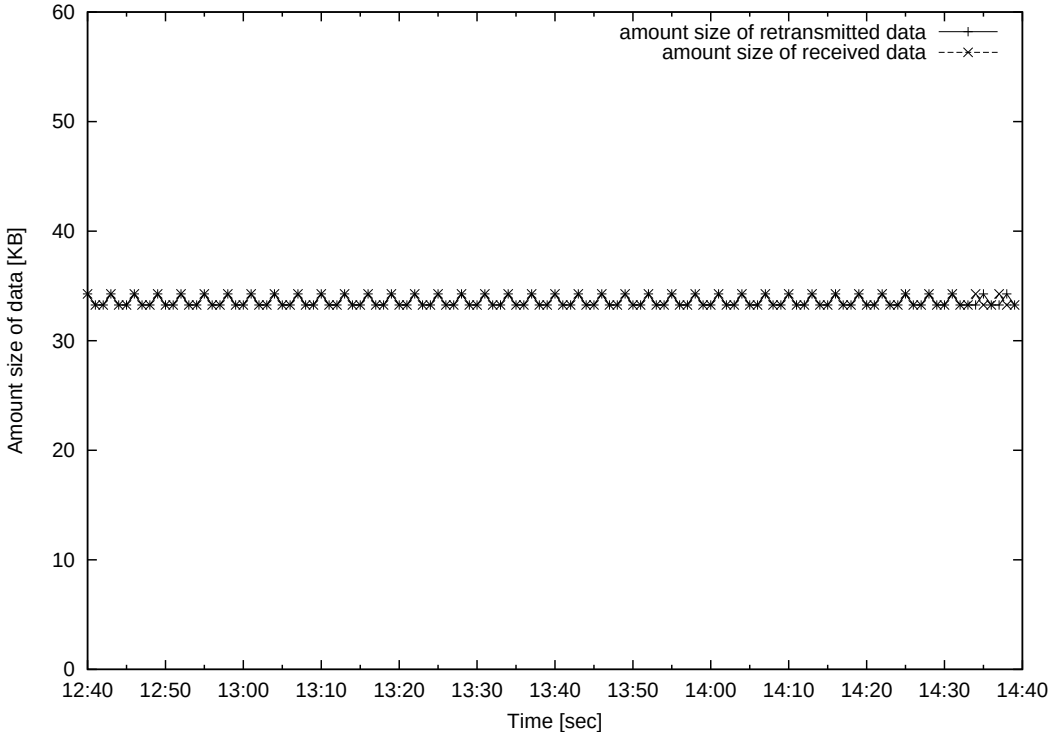

図 B.25: fec0::7 のデータフローに関する R の送受信データ総量/秒の比較 (実験 3)

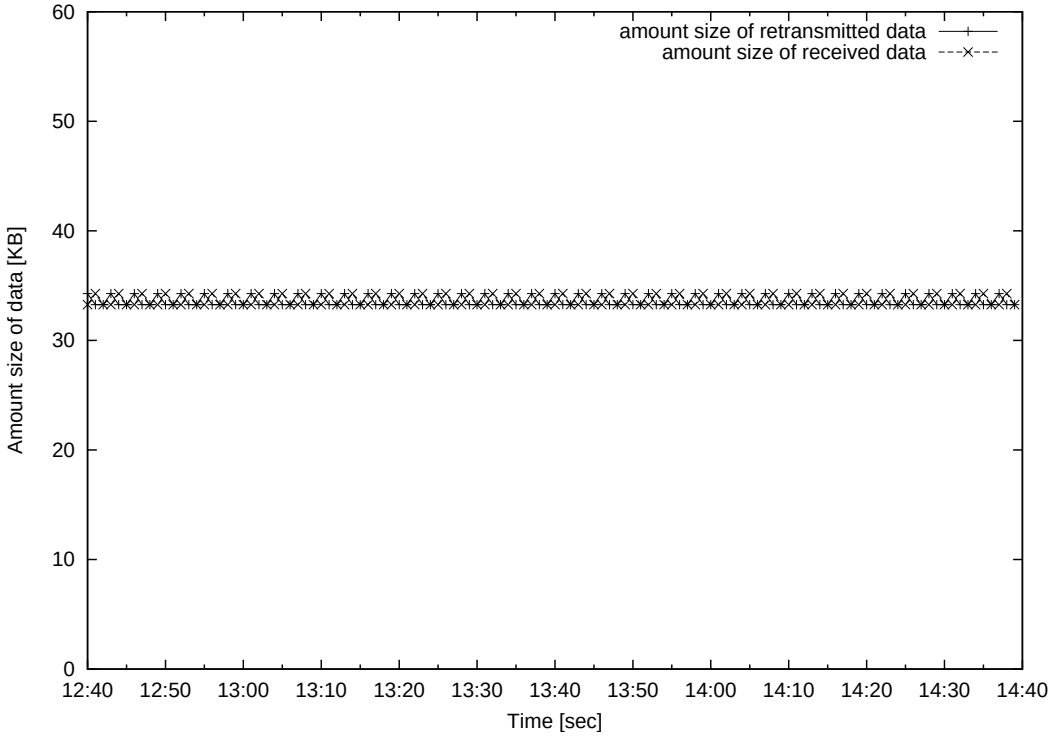

図 B.26: fec0::9 のデータフローに関する R の送受信データ総量/秒の比較 (実験 3)

付 録C　Flooding Messengerの実装

本付録では，Flooding Messenger を紹介する．

C.1　Flooding Messengerの概要

Flooding Messenger は，今後 Adaptive Application Flooding に適用予定のアプリケーションである．Flooding Messenger はその名前の示す通り，メッセージをフラッディングすることで近隣のユーザとのコミュニケーションを実現するアプリケーションである．Flooding Messenger は受信したメッセージを単純に表示するのではなく，メッセージのホップ数に応じて，表示する文字の大きさを変化させるという特徴を持つ．このことを，Flooding Messenger のスクリーンショットを用いて説明する．

図 C.1: Flooding Messenger のスクリーンショット

図 C.1は，Flooding Messenger のスクリーンショットを示す．図 C.1から，各メッセージの大きさが異なることがわかる．これらのメッセージの大きさはそれぞれ，ホップ数に応じて変化している．例えば，画面において，“sora: (^ ^)/” というメッセージが最も大きく表示されているが，これはメッセージ送信者が 1 ホップ以内にいることを意味する．また各メッセージの色もそれぞれ異なっているが，これらの色は複数のメッセージの文字色が重複しないように決定される．各メッセージは一定時間画面内を移動しながら表示され，一定時間が経過すると新しいメッセージに更新される．

C.2　実装環境

Flooding Messenger のプロトタイプは，SFC 環境情報学部 3 年の空閑洋平氏の協力の下，実装した．表 C.1に Flooding Messenger の実装環境を示す．

表 C.1: Flooding Messenger の実装環境

OS	Debian GNU/Linux kernel-2.6.9, Windows XP pro SP2
使用言語	C 言語, Flash MX 2004

C.3　Flooding Messenger の設計概要

次に Flooding Messenger の設計概要を図 C.2に示す．

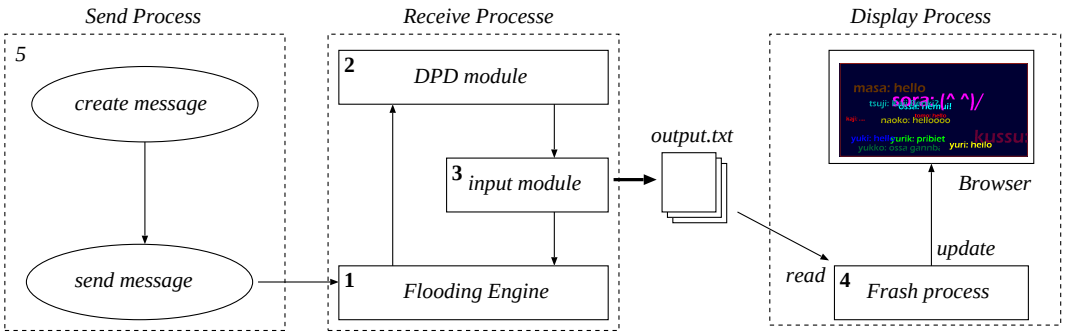

図 C.2: Flooding Messenger の設計概要

1. Flooding Engine
   Flooding Engine は単純なフラッディング (Classical Flooding) を行うモジュールである．Flooding Engine はパケットを受信すると DPD module へパケットのデータ部分を渡す．また，INPUT module から渡されるデータに関しては，ヘッダ内に記述される TTL を参照することにより再送するか否かを決定する．

2. DPD module
   DPD module はデータの重複検知を行う．重複したデータでない場合は，そのデータを input module にデータを渡す．また，DPD module では一定時間以上更新されないデータフロー情報の削除も行う．

3. INPUT module
   INPUT module は DPD module から受信したデータをファイルへ書き出す処理を行う．処理後，データを Flooding Engine に渡す．

4. Frash process
   Frash の処理は，設定ファイルを読み込み，指定されたブラウザへのメッセージ書き込み

の 2 つである．なお，ブラウザは Frash のファイルを開くことにより，自動的に実行される．

5. Send process
   Send process はメッセージを送信する際に利用される．Send process は入力されたメッセージを Receive process に渡す処理のみを行う．

図 C.2に示したように，Flooding Messenger は 3 つのプロセスにより実現される．これらのうち，Display process に関しては，Flash に記述される処理が担当する．また，Flooding Messenger を Adaptive Application Flooding に適応するための要件は，Send process のメッセージ作成部分，Receive process の INPUT module の 2 点を再度実装することである．しかし，これらの処理は図 C.2に示すように非常に単純な動作のみを行う．したがって，Flooding Messenger は容易に Adaptive Application Flooding に適用可能である．

C.4 動作実験

Flooding Messenger は SFC 環境情報学部の教員である湧川隆次氏が担当する授業，“情報技術ワークショップ” にて約 20 名の生徒に利用された．図 C.3 に示すように，動作実験は SFC のキャンパス内で行った．

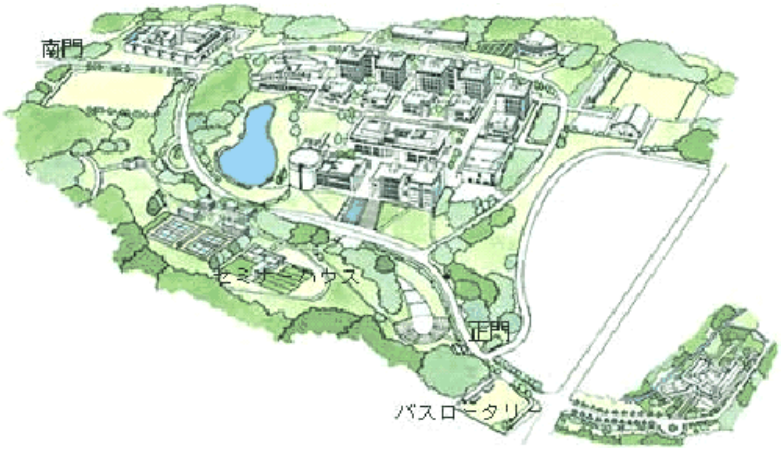

図 C.3: 動作実験環境

動作実験は 20 分の時間を設け行われた．この間，生徒には Flooding Messenger の動作する端末を持つこと，キャンパス内を適当に歩き回ることを指示した．動作実験では，筆者の端末で動作している Flooding Messenger のスクリーンショット，スクリーンキャプチャを撮り，また実験の様子は DV カメラを利用し収録した．図 C.4及び図 C.5は，動作実験中に撮った Flooding Messenger のスクリーンショットである．

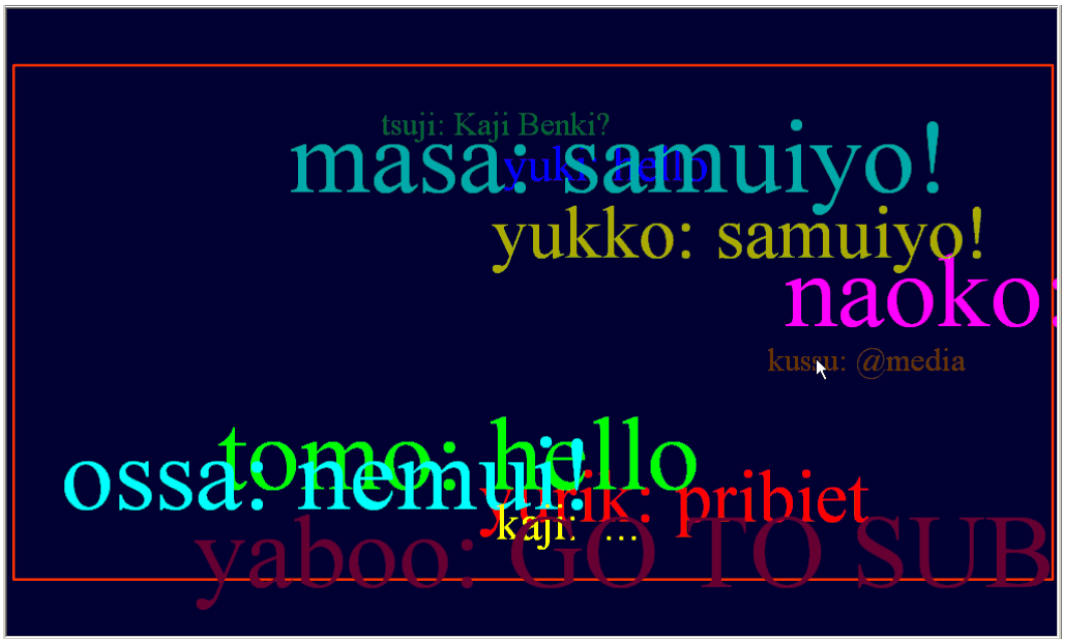

図 C.4: 動作実験時におけるスクリーンショット 1

図 C.5: 動作実験時におけるスクリーンショット 2