平成 25 年度　特別研究報告書

アドホックネットワークにおける P2P ストリーミング環境の構築

龍谷大学　理工学部　情報メディア学科

T100405　田中　雄也

指導教員　三好　力　教授

内容概要

近年のストリーミング配信では,サーバ・クライアント方式と P2P 方式の二つの配信方式が利用されている.しかし,これらすべては,インターネット環境を用いて行われており,災害などによってインターネットが利用できない場合に,ストリーミング配信を行うことができない.そこで,インターネット環境ではなく,アドホックネットワーク環境で P2P ストリーミングを構築することによって,災害発生時など,インターネット環境を利用できなくなった場合に,現地の情報を多くの人に配信することができると考えた.アドホックネットワークとは,インターネットを利用せず,無線を用いて端末同士が直接通信を行い,ネットワークを構築していく方法であり,無線が届く範囲の中にある端末を経由して目的の端末まで通信網を確立していくことである.これを利用することによって,インターネット環境が利用できない場合でも,ストリーミング配信を行うことができると考える.本研究では,アドホックネットワーク環境に P2P ストリーミングを構築する際に,パケットを効率よく配信する手法を提案する.

目次

第 1 章　はじめに

1.1　研究背景

近年,インターネット環境を利用し,動画などのマルチメディアを誰とでも共有することのできる社会となっている.その中で,インターネットを利用し,youtube やニコニコ動画,Yahoo 動画などの映像をいつでも視聴することができるストリーミング再生が普及している.また,スマートフォンや携帯電話の普及によって,時間や場所によらず,どこでも視聴できるようになり,より多くの人々に使われるようになっている.また,youtube,ニコニコ動画などの動画配信コンテンツでは,誰でも視聴できるだけでなく,自分が所有している映像などのコンテンツを他人と共有することもできる.ストリーミング専用サーバにデータをアップロードすることによって,誰でも視聴することができるため,東北大震災では,現地の状況などを住民から全国へと情報発信を行っていく一つのツールとして利用されていた.

動画再生の既存技術としては,動画データなどのコンテンツをクライアントが要求し,ダウンロードすることによって,動画再生を行うことが主流であった.ダウンロードが終了してから再生することによって,再生においての遅延などがなくスムーズに再生ができるという特徴がある.

また,ストリーミング再生では,動画データをダウンロードしながら再生を同時に行うという方法であり,データを保存せず,再生後に削除することによって使用する機器自体の容量を気にすることなく映像の視聴を行うことができる.しかし,問題点もあり,再生時間までにパケットが届かない場合,映像に途切れや遅延が生じてしまい,品質が低下してしまう.この問題点を改善するために多くの研究が行われており,多くの通信方式などが考えられている.

ストリーミング再生では,一般的にサーバ・クライアント方式を用いてコンテンツをストリーミング専用サーバにアップロードし,ストリーミング方式を用いてクライアントが視聴する方式であるが,この方式では,サーバがコンテンツの送受信をすべて行っていくため,サーバに負荷が大きくかかってしまう.また,サーバと接続するクライアント数が多くなることによって,他のクライアントがサーバに接続できなくなり,待ち時間や遅延が生じてしまう.そこで,サーバ・クライアント型の一極集中型ではなく,分散型である P2P ネットワークが近年ストリーミング再生において注目されている.

P2P ネットワークは,クライアントがサーバとクライアントの両方の性質を持ち,クライアント同士でデータの送受信などを行う方式である.P2P を用いたストリーミングにおいては,サーバ・クライアントと同じくコンテンツを分割し送受信を行うが,サーバから分割データを受信するのではなく,P2P ネットワークに参加しているピアの中から受信したいピースを持っているピアを選択し,受信する.これによって,ネットワーク内で一つのピースを拡散していき,一つのピアに対して受信要求が集中し,通信帯域が狭くなってしまうことを防ぐこと,また,負荷をネットワーク全体で分散することができる.しかし,P2P を用いたストリーミング再生において問題点があり,ピアの通信帯域や受信先のピアがネットワークから離脱することで,他のピアからデータを受信するなどの問題点が生じることによって,再生時間までに次の再生データが間に合わず,待ち時間や乱れが生じる.

近年,この問題点を改善するための研究が多く行われており,その一つに,ネットワーク内でパケットを効率よく拡散させ,多くのピアでコンテンツにおける複数のパケットを共有することで,ストリーミングにおける再生の途切れを減少させていく研究がある.この研究は,P2P ネットワーク環境で行われており,インターネット環境を介して,ピア同士が直接パケットの送受信を行っている.

また,パソコンだけでなく,スマートフォンなどの携帯端末を利用して音楽やラジオなどのコンテンツをストリーミングを用いて利用するユーザーが増加している.2013 年 2 月時点で 94 万人がスマートフォンでストリーミング再生を利用しており,前年度比で 71%増加している.

しかし,パソコンやスマートフォンなどでストリーミングを利用する際には ,必ずインターネット環境を利用してストリーミング再生を行っている.もし,災害などによってインターネット環境に接続することができなくなった際に,災害の情報や映像などをストリーミングを使って発信することができなくなる.そのような状況になった際,アドホックネットワークを用いることによって情報の共有などを行うことができる.アドホックネットワークにおけるパケットの送受信の仕組みとしては,端末がパケットを送信する際に,直接届けたい端末に接続して送信するのではなく,他の端末を経由していき,無線通信を用いて相手端末へと送信する.このように,他の端末を経由して,相手端末へと送信することによって自由なネットワークを構築することができ,途中の端末で問題点が起こり,通信が行えなくなった場合でも,他の経路を構築することによって通信を行うことができる.これらから,インターネット環境を利用できなくなった場合に,P2P をアドホックネットワーク環境で構築していくことによって,情報を広く拡散することができると考える.

1.2　研究目的

上記のように,インターネット環境での P2P ストリーミング再生を,アドホックネットワーク環境で用いることによって災害などにおいて有効に活用できる.しかし,インターネット環境での P2P ストリーミング再生を,アドホックネットワークの環境に実装する際に,既存技術を用いての実装では,効率よくパケットの送受信を行うことができない.そこで,アドホックネットワークの特徴を活かしたストリーミング再生における手法を提案する.

アドホックネットワークでは,受信ピアが,欲しいピースを持っているピアに対して直接送受信を行うのではなく,他のピアを経由してピースの送受信を行っていくため,他のピアがピースを経由している際に,経由しているピアがピースを記憶していくことによって,ネットワーク内におけるピースの拡散を効率よく行っていくことができると考えられる.そこで,上記のようなアドホックネットワークの特徴を考慮し,アドホックネットワークの環境におけるストリーミング再生の途切れなどを軽減するパケットの受信方式について考え,研究を行う.

1.3　各章の構成

第 1 章では,研究背景について説明を行う.第 2 章では,既存手法を説明する.第 3 章では,既存手法における問題点の定義と提案手法について説明を行う.第 4 章では,本実験結果を記載する.第 5 章では,考察を行う.第 6 章では,まとめについて記載する.

第 2 章　P2P ネットワークにおける既存技術

近年,ストリーミング配信は,パソコン端末のみでなく,スマートフォン・携帯電話・タブレットなど,時間や場所に依存せず,どこでも動画データなどのマルチメディアの共有,視聴などを行うことができるようになっている.また,パソコンにおけるストリーミング再生においても研究が多く行われており,サーバに動画データをアップロードしておき,クライアントがサーバから直接動画データをもらうサーバ・クライアント型の方法だけでなく,P2P を用いたクライアント同士でのストリーミング配信が研究されている.P2P ストリーミング配信は,二つの P2P ネットワークを利用する方法があり,ピュア型 P2P ネットワークとハイブリッド型 P2P ネットワークの二つに分類される.今回は,既存手法に用いられているハイブリッド型 P2P ネットワークのみ説明を行う.

2.1　ハイブリッド型 P2P ネットワーク

ハイブリッド型 P2P ネットワークを用いて行うストリーミング配信では,P2P ネットワーク内におけるコンテンツ情報を一括して管理を行うサーバが存在し,コンテンツを受信したいピアは,サーバに情報を問い合わせることによって,そのコンテンツを所持している接続するべきピアの IP アドレスを取得し,その IP アドレスのピアに接続することで,コンテンツを取得する.また,ハイブリッド型 P2P ネットワークを利用したソフトウェアとして BitTorrent が主にあげられる.

2.2　BitTorrent

BitTorrent とは,ハイブリット型 P2P ネットワークと同じ構成で作られており,トラッカと呼ばれるピアの情報を管理しているインデックスサーバとピアによって,P2P ネットワークを構築しているソフトウェアである.

BitTorrent では,ピアは P2P ネットワークに参加する際に,Web サーバへと接続し,トラッカのアドレスや配布ファイルの情報を含むトレントファイルを取得後,取得したアドレスからトラッカに接続することで,P2P ネットワークに参加する.ピアは,P2P ネットワークに参加している間,自身が所有しているファイル,送信速度,受信速度などの情報を定期的にトラッカへと報告を行う.報告後,他のピアの情報をトラッカから取得し,取得した情報を元に欲しいファイルを持っているピアを選択し,TCP で各ピアと接続する.次に,ピアの接続と受信に関する選択ポリシーを説明する.

2.2.1　接続ピアの選択ポリシー

BitTorrent では,ピアの個数に関係なく,複数のピア同士で同時にコンテンツを送受信することができる.しかし,一つのピアに対して複数の要求が集中してしまった際に,通信帯域が狭くなり,速度が低下してしまう.この問題の解決として,受信速度の向上のために,ピアが接続するピア数を一定数に制限し,受信ピアの通信帯域を一定量保つようにしてい

る.BitTorrent では,ピアが送受信を同時に行う数として,4 つまでと制限している.ピアは,トラッカから得た情報を元に,通信速度の速いピアを選択し,現在接続しているピアとの通信速度より速いピアが存在する場合,接続先を変更し,ファイルの送受信を行う.また,接続しているピアから一定時間送信されてこない場合,接続先を変更し,以降そのピアとの接続を行わないようにする.

2.2.2　受信ピースの選択ポリシー

BitTorrent では,ファイルの送受信を行うとき,ピース単位に分割し,送受信を行う.ピースを受信しているときは,ピースの完成を最優先に行い,完了後,ピースの送信を行う許可をする.ピアは,ピース単位で受信することによって,ネットワーク内にピースを拡散させる.受信する際に,接続しているピアからピースを選択する方法としては,レアレスファスト方式を用いる.レアレスファスト方式では,P2P ネットワーク内に存在するピースの個数が最も少ないピースを選択し,受信する方式である.これによって,P2P ネットワーク内で希少価値の高いピースを減少させていき,ネットワーク内での通信状況,待ち時間,通信帯域,受信速度などの向上を行う.

2.3　BiToS

BiToS(Enchancing BitTorrent for Supporting Streaming Application)は,BitTorrent を用いて,P2P ネットワークにおけるストリーミング再生の途切れ時間を減少させる方法である.P2P ネットワークから受信したピース以外のまだ所持していないピースを優先セット,低順位セットに分ける.優先セットは,再生時間に近いピースを固定長分選択したセットであり,低順位セットは,それ以外のピースで構成されている.ピアがピースの受信を選択する際に,優先セットか低順位セットかを確率で選択する.どちらのセットから選択するかを確率で選択後,選択したセットの中からレアレスファスト方式で選択するピースを決定する.ただし,レアレスファスト方式でピースを選択する際に,セットの中に最も希少であるピースが複数存在する場合,その中でも再生順に一番近いピースを選択する.優先セットからピースを一つ選択し,受信を行った後,低順位セットの再生順に一番近いピースを優先セットへと移動する.これにより優先セットの中のピースの個数を固定長に保つ.以下に BiToS 方式を表した図を示す.

□:所持していないピース
■:所持しているピース

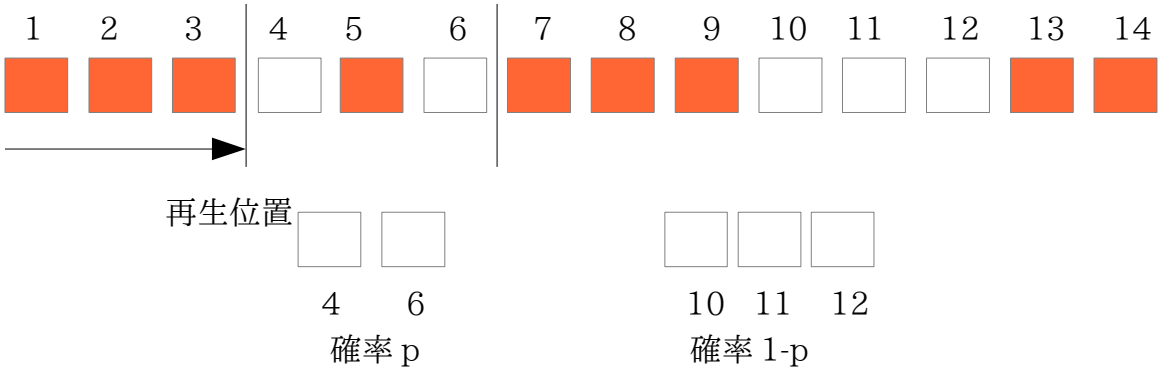

図 1:優先セットと低順位セットの選択方法

2.3.1　問題点

BiToS における問題点としては,優先セットを固定長で保ち,ピースを選択する際にレアレスファスト方式で選択するため,再生に一番近いピースが受信されず,再生時間に間に合わないことがある.

2.4　IS 方式

P2P ネットワーク内で数の少ないピースを受信しようとする際に,そのピースを所持しているピア数が少ないため,一つのピアに受信要求が集中してしまう.一つのピアに複数の受信要求が同時に起こることで,通信帯域が狭くなり,受信速度が遅くなってしまう.上記のことから,ピースを選択する順序はかなり重要となってくることが考えられる.そこで,IS(Immediacy and Scarcity)方式では,受信ピースを選択する時に,全ピースに対する重要度を算出する.重要度の算出は,希少性と緊急性の二つを考慮し,算出する.これにより算出された重要度の高いピースを選択し,受信を行う方式である.重要度を算出する式は,以下の式となる.

$$Di = c \cdot Ii + (1 - c) \cdot Si \qquad (1.1)$$

この式において,Di は i 番目のピースの重要度を表す.Ii は i 番目のピースの緊急性を表し,再生順に近いピースほど緊急性が高くなる.Si は i 番目のピースの希少性を表し,P2P ネットワーク内におけるピースの個数が少ないほど希少性の値が高くなる.c は重み係数を表し,c の値が高いほど緊急性を重要視するようにし,再生順に近いピースを選択し受信するようにする.また,c の値が小さいほど希少性を表し,ネットワーク内に個数の少

ないピースを受信し,ネットワーク内での個数を増加させるように受信する.重要度を選択し,最も高い数値が複数あるときは,再生順に近いピースを選択する.次に,緊急性と希少性の式について以下に説明する.

2.4.1　緊急性 Ii

緊急性は,再生位置から近いピースほど高くなり,遠いピースほど小さくなる.再生位置は,P2P ネットワーク内に参加しているピアごとによって変化し,各ピアごとで緊急性の数値が変化する.これらから,緊急性の式は以下となる.

$$Ii = 0 \qquad (i = 0, \cdots\cdots\cdots, h - 1)$$

または,

$$Ii = 1 - (i - h) / l \qquad (i = h, \cdots\cdots\cdots, l) \qquad (1.2)$$

この式において,h は次に再生されるピースを表す.l は,そのコンテンツをパケット単位に分解したときの最後のピースを表す.

2.4.2　希少性 Si

希少性は,P2P ネットワークの中に存在する同じピースが多いか少ないかを表す.Si が大きくなるほどネットワーク内に存在するピースが少ないこと表し,大きくなるとそのピースがネットワーク内に多く存在していることを表す.希少性が小さいということは,ネットワーク内に存在しているピースの個数が少ないことを表しているので,待ち時間や通信帯域が狭くなってしまう問題が生じる.これらより,希少性を以下の式で表す.

$$Si = (N - m) / N \qquad (1.3)$$

この式において,N は P2P ネットワークに参加しているピアの全体個数を表す.m はピース i を保持しているピアの数を表す.そのピースを所持しているピア数が大きくなるほど m の値が大きくなり,N - m の値が小さくなることで,Si の値が小さくなる.よって希少性が低いと表すことができる.

2.4.3　問題点

IS 方式は,ピースに対して重要度を設定し,緊急性と希少性を考慮してピースを受信していく.しかし,コンテンツが提供されて間もない場合,ネットワーク内にはピースがまだ拡散されていないか数が少ないため,オリジナルコンテンツを所持しているサーバ,または,ピアに受信要求が集中してしまい,受信速度が低下してしまう.それによって,ピースを受信する際に待ち時間が生じ,再生が途切れてしまう.

2.5　BIS

上記で記述しているBiToS方式とIS方式の両方には,問題点がともに存在している.この問題点を改善するために,二つの方式を合わせた分割データ受信方式がBIS(BiToS + IS)方式となる.BIS方式では,はじめにコンテンツにおける全ピースに対してBiToS方式で用いられている優先セットと低順位セットの二つに分類する.優先セットが確率pで選択された場合,優先セットの中からピースを選択する際にIS方式を用いてピースの重要度を設定し,重要度の高いピースを選択し受信する.低順位セットが選択された場合は,レアレスファスト方式を用いてピースを選択する.これは,低順位セットのピースは,再生を行うまでに時間があり,緊急性が低くなるため考慮せず,希少性のみを考慮するレアレスファスト方式を用いることが適切と考えられるためである.

第 3 章　既存手法の問題点と提案手法

3.1　問題点の分析

既存手法における問題点とインターネット環境で行っている既存手法をアドホックネットワーク環境で行う際に起こる問題点を以下に示す.

3.1.1　既存手法の問題点

既存手法では,P2P ネットワークに参加する際に,インデックスサーバの情報を得てからネットワークに参加する.インデックスサーバに接続後,ストリーミングを開始する場合,接続先ピアなどの情報をインデックスサーバから取得してストリーミングを行う.しかし,同時に多くのピアからインデックスサーバへと受信要求を送ることによってインデックスサーバの通信帯域が狭くなり,遅延やパケット損失が生じてしまう問題点がある.この問題は,アドホックネットワーク環境ではより深刻であると考えられる.

3.1.2　既存手法をアドホックネットワークに実装する際に生じる問題点

既存手法のネットワーク環境では,P2P ストリーミングをインターネット環境で実装し,ピースの共有を行っていた.しかし,アドホックネットワークなどピア同士が直接通信を行わない場合,他のピアを経由し,目的のピアにデータの送信を行っていくため,一つのデータを送信する際に,直接相手ピアに対して送信することができない.また,アドホックネットワークでは,携帯端末など移動性を持つ端末との通信などネットワーク自体を自由に構築することができる.しかし,ピア自身が移動性を持ち,ネットワークを自由に構築することによって,ピアのネットワークに対する出入りが頻繁に生じることがあり,ピースの損失や通信の切断が起きる問題点がある.

3.2　提案手法

3.2.1　提案手法により改善する問題点

上記の問題点の中で,提案手法により改善を試みる点を以下に示す.

問題点 1　同時に多くのピアがインデックスサーバへと情報の送信要求を行うことによりインデックスサーバの通信帯域が狭くなり,遅延やパケット損失が起こってしまうこと

問題点 2　インターネット環境で構築されている既存手法をアドホックネットワーク環境で構築する際に中継ノードを考慮していないこと

上記の 2 点の問題点を改善する提案手法を以下に示す.

3.2.2　経由しているノードの活用

一つ目のとして,問題点２を改善する提案手法を示す.

ピアがアドホックネットワーク環境でピースの送受信を行う際,ピア同士が直接リンクを接続し送受信するのではなく,他のピアを経由して,ピースの送受信を行う.その中継ピアがピースを経由した際に,図２のようにピースを記憶させてから次のピアへと送信する.これを行うことによって,ホップ数が大きくなるほどピースを記憶する中継ピアが増加する.しかし,既存手法では,ストリーミングを開始し,受信要求を行ったピアのみがピースを記憶するため,ネットワーク内のピースの個数は一つのコンテンツ分のピース枚数しか増えない.よって,提案手法の中継ピアがピースを記憶することにより,ホップ数の数だけ一回の送信でネットワーク内にピースの個数を増やすことができ,同じ再生回数でも,既存手法よりも多くピース総数を増加させることができると考える.例えば,送信ピアから受信ピアまでのホップ数が４であるとき,経由するピアは３つとなるので,一回の送信によって４つのピアがピースを記憶し,ネットワーク内でピースを拡散することができる.しかし,経由するピア自身は,ストリーミング再生を行わないのにも関わらず,経由する度にピースを記憶していくことで,大きな容量を使ってしまう問題点が生じる.そこで,経由するピアは,ピースを中継する際に記憶するピースの個数をＮ個までとし,新たにピースを経由する場合,3.2.3 節の拡張リング法によって構築したネットワーク内で最も希少性の低いピースを破棄し,新たに記憶する.これによって,中継ピアが記憶する容量を一定値に保つ.

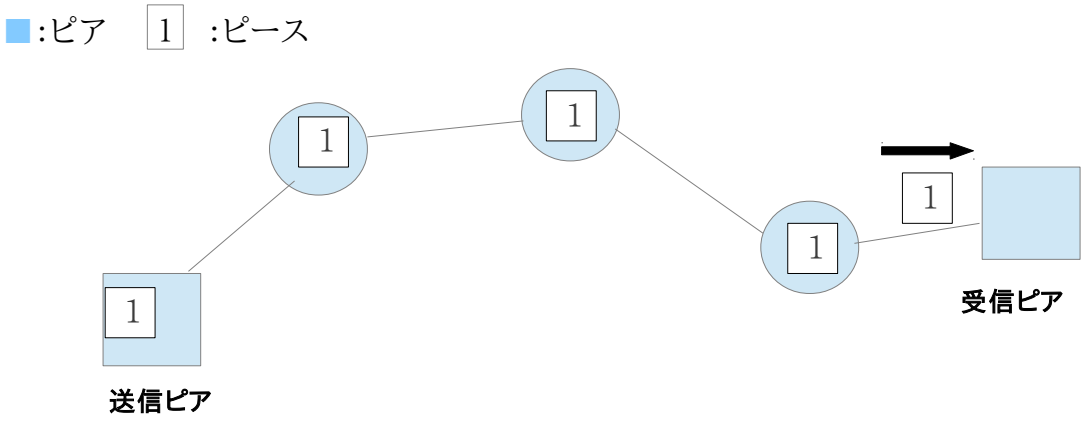

図 2:アドホックネットワークのパケット移動

3,2,3　インデックスサーバを使用せずピアによって経路探索の決定

二つ目として,問題点１を改善する提案手法を示す.

既存手法では,有線による直接通信を前提にインデックスサーバから情報を取得して,接続先のピアを選択しピア同士の通信を行う.しかし,アドホックネットワーク環境では,インデックスサーバから情報を得る際に,他のピアを経由しなければならない場合があるため,

情報を取得したいピアからインデックスサーバまでのホップ数が大きい場合,情報を取得するまでに時間が多くかかること,また,ホップ数が増加することによってパケットを損失することやリンクが切断してしまう可能性が大きくなってしまう問題点もある.また,問題点 1 でもあるように,アドホックネットワーク環境においても,受信要求を行うピアが複数ある場合,インデックスサーバに接続先などの情報を取得するための要求が集中してしまう.

そこで,アドホックネットワーク環境で P2P ストリーミング再生を行う際に,まず受信要求を行うピア自身がインデックスサーバとなり,MANET では通信路形成のために必ず行うフラッディング時に,ネットワーク内のピアがどのピースを所持しているかなどのピアの情報を取得することを考えた.フラッディングとは,目的となるノードを探すまで経路探索を行う手法で,図 3 のように,発信ノードが経路探索を行うパケットをブロードキャストする.そのパケットを取得したノードは,自身が目的となるノードではない場合に中継ノードとなり,ブロードキャストする.この動作を目的のノードが見つかるまで繰り返していくことで経路探索を行う.そこで,このフラッディングの動作を拡張し,フラッディングでパケットを受け取ったノードは,発信ノードに対し自身が持っているピースなどの情報を送信する.これをサーバを見つけるまでにパケットを受け取ったノードすべてが行い,発信元のノードが受け取ることによって,発信元のノード自身がインデックスサーバの役割を果たし,インデックスサーバを使わずネットワーク内のノードの状況を把握することができる.

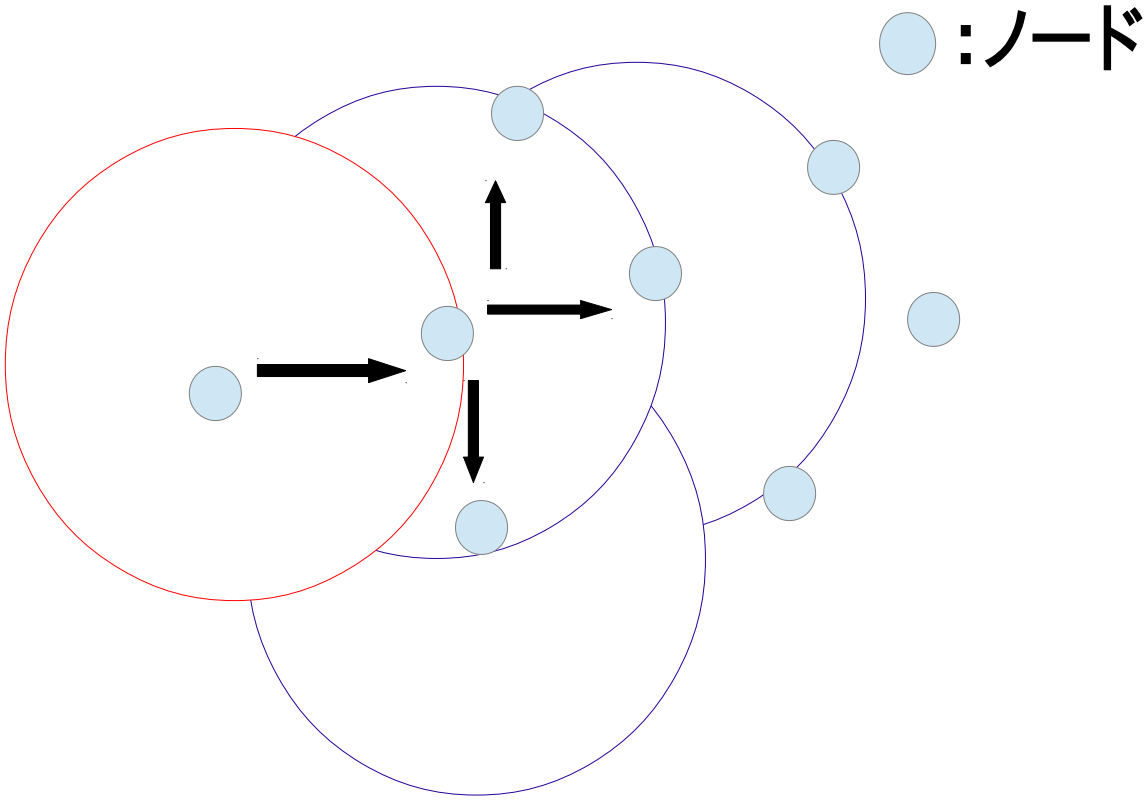

図 3:フラッディングの概念図

3.3　ピース取得の流れ

3.3.1　既存手法におけるピース取得の流れ

既存手法として,BIS 方式を用いた P2P ストリーミングを使用する.以下にストリーミング開始から終了までの流れを示す.

① インデックスサーバの IP などの情報を所持しているサーバに接続し,インデックスサーバの情報を取得する

② 取得した情報からインデックスサーバへと接続し,ストリーミングを行う P2P ネットワークに参加する

③ ストリーミングを開始するピアは,IS 方式を用いて優先セットと低順位セットにピースを分割し,確率によってどちらのセットからピースを取得するかを選択する.

④ 取得するセットを選択後,優先セットからピースを取得する場合,BiToS 方式を用いて最も重要度の高いピースを選択し,取得する.その後,6 へ進む

⑤ 低順位セットからピースを取得する場合,希少性を重要視し,再生時間に余裕があるため,レアレスファスト方式を用いてピースを選択し,受信する.その後,6 へ進む

⑥ ピースの受信を確認後,受信したピースを他のピアへと送信する許可を行い,要求があれば送信を行うようにする.その後,3 の動作に戻り,再生に必要なピースを取得する

3.3.2　既存手法をアドホックネットワーク環境で構築した場合のピース取得の流れ

既存手法の BIS 方式をインターネット環境からアドホックネットワーク環境で構築した時のストリーミング開始から終了までの流れを以下に示す.

① ストリーミングを行うピアがインデックスサーバと接続を行うために、フラッディングを行い、インデックスサーバまでの経路探索を行う.

② インデックスサーバを見つけた場合、ストリーミングを行う P2P ネットワークに参加し、ネットワーク内に存在するピアの情報とオリジナルコンテンツを取得しているサーバの情報を取得する.

③ インデックスサーバからネットワーク内の情報を取得後、もう一度フラッディングを行い、オリジナルコンテンツを所持するサーバまでの経路探索を行う.

④ オリジナルコンテンツを所持しているサーバを見つけた場合,そこで経路探索を終了する

⑤ 経路探索終了後,再生を行うピアは,IS 方式を用いて優先セットと低順位セットにピースを分割する.その後,確率によってどちらのセットからピースを取得するかを選択する

⑥ 取得するセットを選択後,優先セットからピースを取得する場合,再生位置に近いピースを表すので,ピースを正確に受信する必要性が高くなる.そこで,オリジナルコンテンツを所持しているサーバが経路探索によって見つかっている場合,サーバからピースを受信する.ピースは,BiToS 方式によってどのピースを取得するかを選択する.その後,8 の動作を行う.

⑦ 低順位セットからピースを取得する場合,希少性を重要視し,再生時間に余裕があるため,正確性を考慮せず,ピースを所持しているピアからピースを受信する.

⑧ ピースの受信を確認後,受信したピースを送信する許可を得て,要求があれば送信を行うようにする.その後,5 の動作に戻り,再生に必要なピースを取得する.

3.3.3　提案手法におけるピース取得の流れ

提案手法は,既存手法の BIS 方式を改良して作成した,ストリーミングの開始から終了までの流れを以下に示す.

① 提案手法では,インデックスサーバを使用しないため,どのピアが情報を持っているかをストリーミングを行うピアが知らないため,まず,ストリーミングを行うピアが,フラッディングを行い,まわりのピアがどのピースを持っているか,また,オリジナルコンテンツを所持しているサーバを見つけるまで経路探索を行う.

② オリジナルコンテンツを所持しているサーバを見つけた場合,そこで経路探索を終了し,4 の動作を行う.

③ サーバが見つかっていない状態で,経路探索を行ったピアからコンテンツのピースをサーバ以外のピアからすべて取得することができる場合は,そこで経路探索を終了し,4 の動作を行う.

④ 経路探索終了後,再生を行うピアは,IS 方式を用いて優先セットと低順位セットにピースを分割する.その後,確率によってどちらのセットからピースを取得するかを選択する

⑤ 取得するセットを選択後,優先セットからピースを取得する場合,再生位置に近いピースを表すので,ピースを正確に受信する必要性が高くなる.そこで,オリジナルコンテンツを所持しているサーバが経路探索によって見つかっている場合,サーバからピースを受信する.ピースは,BiToS 方式によってどのピースを取得するかを選択する.その後,7 の動作を行う.

⑥ 低順位セットからピースを取得する場合,希少性を重要視し,再生時間に余裕があるため,正確性を考慮せず,ピースを所持している一番ホップ数の少ないピアからピースを受信する.その後,7 の動作を行う

⑦ ピースの受信を行う際に,経由しているピアは,経由したピースを記憶し,ネットワーク全体でのピースの個数の増加を行う.また,経由するピアが,ピースを所持することができる個数を N 個と限定し,再生を行わないピアの記憶容量の増加を低減する.N 個所持しているピアが新しくピースを経由する際,所持しているピースの中で最も希少性の低いピースを破棄し,新たに記憶する.

⑧ ピースの受信を確認後,受信したピースを送信する許可を得て,要求があれば送信を行うようにする.その後,4 の動作に戻り,再生に必要なピースを取得する.

以下に既存手法と提案手法の流れをフローチャートで図示し,改良点を示す.

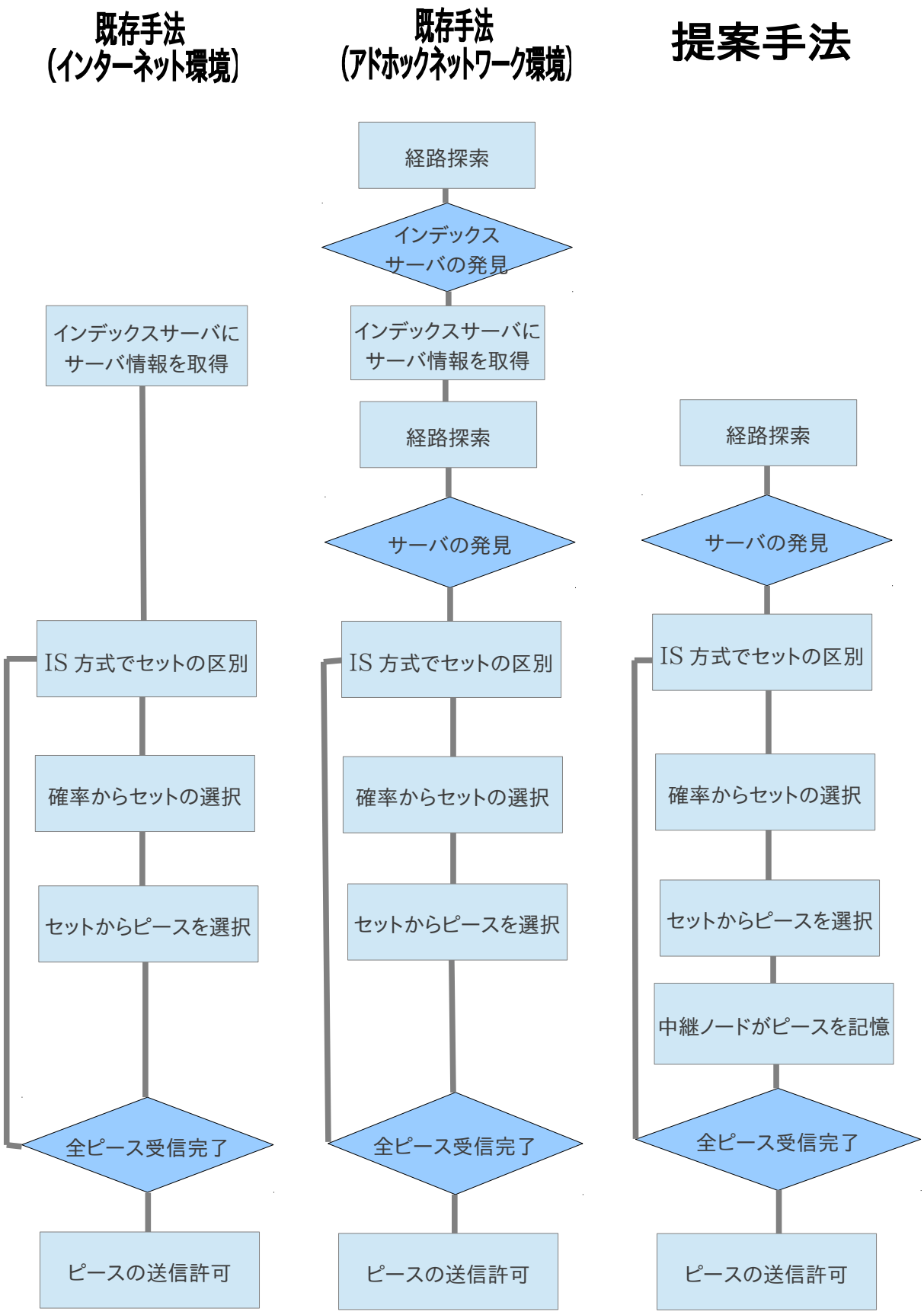

図 4,再生までの流れ(フローチャート)

上記の既存手法と提案手法の流れの違いでは,2 点存在する.一つ目は,既存手法においてはインデックスサーバを使い,ピースを取得するピアの情報を取得し,その情報から相手ピアと接続し通信を行うが,既存手法をアドホックネットワーク環境に構築した場合、インデックスサーバに情報を取得する際に経路探索を行い、インデックスサーバまでの通信経路を構築し、その後、オリジナルコンテンツを取得しているサーバまで接続するために,経路探索を行なわなければならない.ストリーミングを再生するまでに二回経路探索を行なうことになり、時間が大きくかかり、大きな負荷がかかってしまう.そこで、提案手法では、インデックスサーバを使用せずピア自身がインデックスサーバの役割を果たすことによって、経路探索を一回で済ますことができ、時間と負荷を両方とも低減することができると考える.

二つ目は,既存手法では,インターネット環境で P2P ストリーミングを行う際に,ピア同士が直接データの送受信を行うため,中継ノードが存在しない.しかし、既存技術をアドホックネットワーク環境で構築した場合、他のピアを中継して目的のピアまでデータの送受信を行なうため、中継ピアが必要となる.そこで,提案手法では,中継ピアを考慮し,ピースを中継する際にピースを一時的に記憶しておく.これにより、ネットワーク全体で少ない再生回数であっても,既存手法より多くピースを拡散していくことができると考える.

第４章　実験と評価

アドホックネットワークは,データの送受信を行う際,他のノードを経由して目的のノードへとデータを送受信する.そこで,3.2 節で述べた提案手法を用いることで,アドホックネットワーク環境で効率的にピースを拡散させていくことができるかを評価する.

4.1　NS2 を用いてのシミュレーション環境

本実験では,ネットワークシミュレータの NS2 を使用し,P2P ストリーミング環境を構築する.構築したストリーミング環境を用いて,映像コンテンツの配信を擬似的に行い,アドホックネットワーク内でのピースの増加量とコンテンツをすべて取得するまでの時間の 2 点について評価を行う.

4.1.1　NS2 とは

NS は無料のネットワークシミュレータの一種で,カリフォルニア大学バークレイ校で開発され,現在においても改良が続けられている.NS2 は,NS のバージョン2を表し,本実験では,NS2.35 を使用し,実験を行った.NS2 は,Linux 系 OS で動き,C++と Otcl 言語で書かれたオブジェクト指向のイベントドリブン・ネットワークシミュレータでありローカル・広域ネットワークをシュミレートするのに役立つ.既存のモジュールを使用してシミュレーションする際は,Otcl 言語を使用し,また,新たなモジュールを使用する際には,C++で追加することで使用できる.Otcl 言語などを用いて記述したスクリプトに従いトレースファイルや nam を出力する.本実験では,NS2 を用いて P2P ストリーミング環境を構築し実行することで,トレースファイルにピアが行った動作の詳細を記述し出力する.また,トレースファイルを元に実行したシミュレーション内容をアニメーション化するために nam を用いて視覚化する.以下に,トレースファイルと nam における説明を示す.

4.1.2　nam

nam(network animator)は,NS2 が Otcl 言語などで記述したスクリプトに従って出力されたトレースファイルをもとに,シミュレーション内容をアニメーション化するツールである.nam を使用することによってパケットの流れや端末の配置やネットワークの構成を視覚化することができ,再生速度も自由に変更できる.以下に nam ツールによってアニメーション化した図を記す.図 5 では,ノードを三台配置した状態を表している.ノードには,一台ごとにノード ID が割り振られており,図 5 では,0 から 2 までの ID を割り振っている.図 5 の左上にある再生ボタンを押すことで,プログラムが実行され,ノードの動作を視覚的にみることができる.

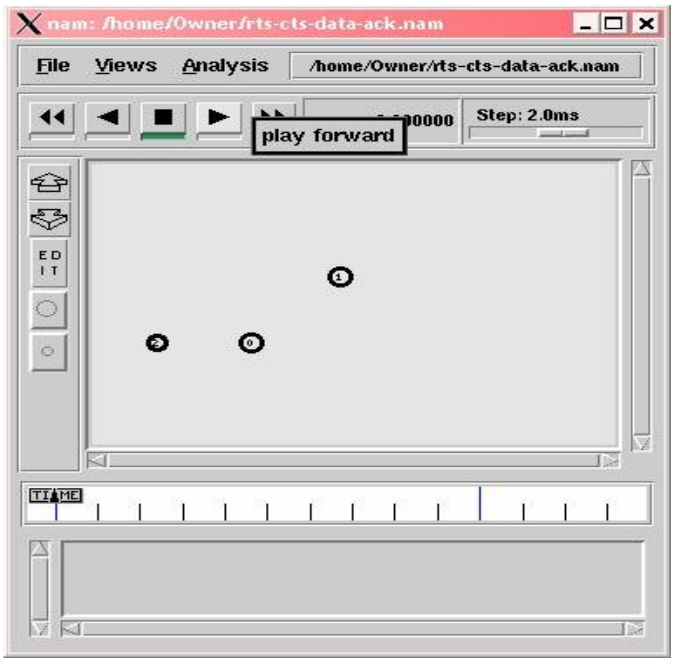

図 5,nam のアニメーション化した図

4.1.3　トレースファイルについて

　NS2 はスクリプトを解釈し,シミュレーションをトレースした状況を出力できる.本実験では,トレースファイルから端末が送受信を行った時間や送信先などを計測し,実験を行った.この NS2 のトレース機能で出力されるファイルが以下の例となる.

s -t 1.000000000 -Hs 0 -Hd -2 -Ni 0 -Nx 100.00 -Ny 200.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 0.0 -Id 1.0 -It tcp -Il 40 -If 2 -Ii 1 -Iv 32 -Pn tcp -Ps 0 -Pa 0 -Pf 0 -Po 0
r -t 1.000000000 -Hs 0 -Hd -2 -Ni 0 -Nx 100.00 -Ny 200.00 -Nz 0.00 -Ne -1.000000 -Nl RTR -Nw --- -Ma 0 -Md 0 -Ms 0 -Mt 0 -Is 0.0 -Id 1.0 -It tcp -Il 40 -If 2 -Ii 1 -Iv 32 -Pn tcp -Ps 0 -Pa 0 -Pf 0 -Po 0

上記の例では,二つのイベントについてトレース状況を表している.イベントは,七組に分けることができる.

・1 組目:イベント
　s が 1 組目にあたり,そのノードが受信しているのか送信しているのかを表す.

・2 組目:イベント時刻
　-t 1.000000000 が 2 組目にあたり,時刻に関する情報を提供する.

・3 組目:次ホップ情報

-Hs 0 -Hd -2 -Ni 0 が 3 組目にあたり,次のホップ情報を表し,現在のノードや目的地に向かうべき次のホップのノード ID などの情報を表す.

・4 組目:ノードの情報

-Nx 100.00 -Ny 200.00 -Nz 0.00 -Ne -1.000000 -Nl AGT -Nw ---が 4 組目にあたり,どのレベルでトレースするか(エージェントかルータか MAC など)の情報を与える.

・5 組目:MAC レベルでのパケット情報

-Ma 0 -Md 0 -Ms 0 -Mt 0 が 5 組目にあたり,接続期間やイーサネットアドレスなどを表す.

・6 組目:IP レベルでのパケット情報

-Is 0.0 -Id 1.0 -It tcp -Il 40 -If 2 -Ii 1 -Iv 32 が 6 組目にあたり,送信元や宛先の IP アドレス,パケットサイズなどパケット情報を表す.

・7 組目:アプリケーションレベルでのパケット情報

-Pn tcp -Ps 0 -Pa 0 -Pf 0 -Po 0 が 7 組目にあたり,アプリケーションタイプの情報を表す.

4.2　実験環境

以下に既存手法と提案手法の実験環境について示す.

4.2.1　既存手法における実験環境

既存手法における実験環境としては,NS2 を使用し,P2P ストリーミング環境を構築した.NS2 で動作するスクリプトとして Otcl 言語を使用し,測定を行った.評価に使用するパラメータとしては,以下の表のとおりとする.

表 1,既存手法におけるパラメータ

パラメータ	値
ピア数	10
再生回数	8
ノード間の距離(X)	250
ノード間の距離(Y)	250
再生開始時間(分)	0.1
再生間隔(分)	1
サーバID	24
コンテンツのピース総数	10

既存手法では,インターネット環境を用いてストリーミングを行う.ストリーミングを行う際に既存手法では,相手ピアの IP などの情報を取得するためにインデックスサーバに接続し,情報を取得しなければならない.これより,本実験では,インデックスサーバをネットワークの中に一つ配置し,すべてのピアはストリーミングを行う際に,そのインデックスサーバに接続し相手ピアの情報を取得するようにした.また,BIS 方式で優先セットと低順位セットに分割する際に,優先セットを 3 枚とする.また,オリジナルコンテンツを所持しているサーバは,ピア ID24 とする.これは,ピア ID24 がネットワーク内で真ん中に位置しているため,実験を行う際に,すべての方向からストリーミングを行うことができるためである.

4.2.2　提案手法における実験環境

提案手法における実験では,既存手法の実験と同じく,NS2 を使用し,P2P ストリーミング環境を構築した.NS2 で動作するスクリプトとして Otcl 言語を使用し,測定を行った.評価に使用するパラメータとしては,以下の表のとおりとする.

表 2,提案手法におけるパラメータ

パラメータ	値
ピア数	49
再生回数	8
ノード間の距離（X）	250
ノード間の距離（Y）	250
再生開始時間（分）	0.1
再生間隔（分）	1
サーバID	24
コンテンツのピース総数	10

ネットワークに参加しているピアは,全部で 49 個配置し,横 250,縦 250 に設定し,ピアの ID として 0～48 までを振り分ける.下記にピアの配置を図 6 に示す.

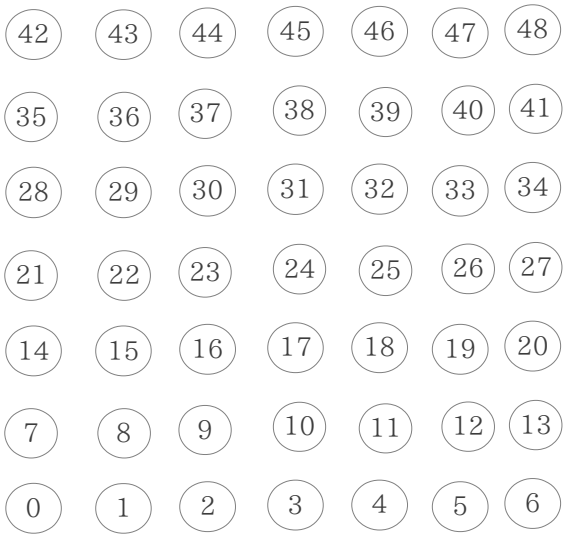

図 6,ピアの配置

上記の図 6 のようにピアをメッシュ型に配置した.これは,ストリーミングを行う際のピア同士の通信の流れをアニメーション化した際に見やすくするためである.この 49 個のピアを用いて P2P ストリーミングを行っていく.本実験では,中継ノードがピースを記憶することができる最大値を 3 枚までと設定する.これは,4.2.2 節で述べた中継ノードがピースを記憶することで大きな容量を使ってしまう問題点が生じないように,3 枚まで記憶することで中継ノードの負荷の低減を行うことができるためである.次に,BIS 方式での優先セットの個数は,既存手法での実験環境と同じく 3 枚とする.また,オリジナルコンテンツを所持しているサーバは,ピア ID24 とする.これは,ピア ID24 がネットワーク内で真ん中に位置しているため,実験を行う際に,すべての方向からストリーミングを行うことができるためである.本実験の流れとしては,3.3 節で述べた提案手法におけるピースの流れにそって行なっていく.

4.3　実験 : ピースの量の比較

中継ノードの活用によってネットワーク内のパケット総数の増加量の変化について実験結果を以下に示し,問題点の改善ができているかについて評価を行う.

4.3.1　アドホックネットワーク内でのパケットの総数

提案手法におけるアドホックネットワーク環境の中継ノードを考慮し,ピアがピースを中継する際に記憶していくことによって既存手法のインターネット環境での P2P ストリーミングを行なうより,同じ再生回数で多くのピースをネットワーク内に拡散させていくことができることを測定するために,既存手法と提案手法において再生回数の増加に伴うネットワーク内のピースの増加量を計測した.評価方法としては,同じ再生回数でネットワーク内に存在するピースの総数が多いほど効率よくピースを拡散することができていることを表す.以下にインターネット環境での P2P ストリーミング再生とアドホックネットワーク環境での

P2P ストリーミング再生の再生回数の増加に伴うピースの総数を図 7 に示した.

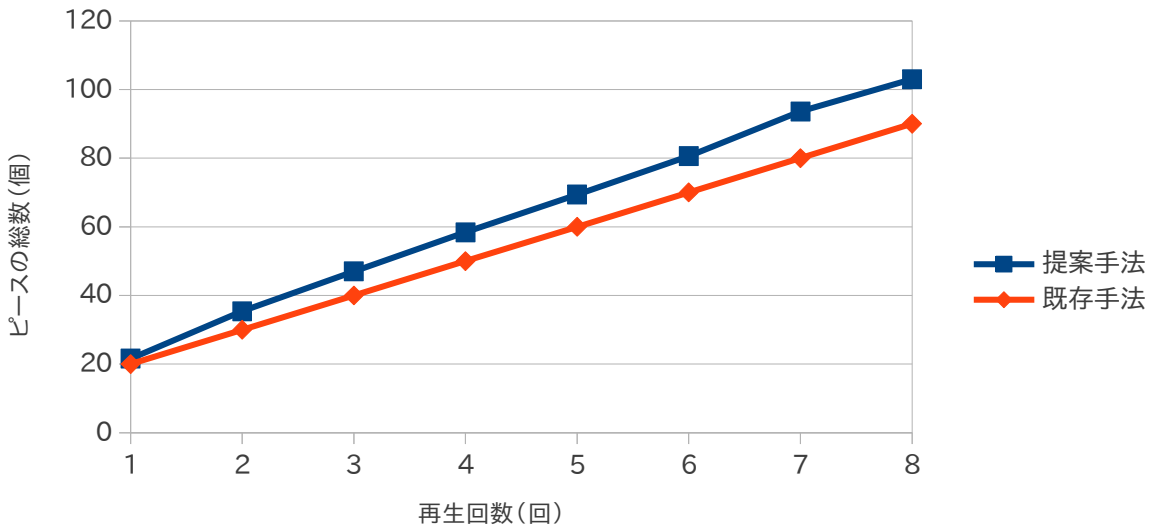

図 7, 再生回数の増加によるピースの総数

図 7 より,従来のインターネット環境で行った P2P ストリーミングでは,ピースの総数は（再生するピア数）*（コンテンツ再生までのピースの総数）個となり,本実験では,オリジナルコンテンツの総ピース枚数を 10 個としているため,一回の再生で 10 個ずつネットワーク内に拡散していくことがわかる.提案手法では,サーバまでのホップ数が 1 となるピア以外では,中継ノードが存在するため,中継ノードが記憶する分のピースが従来の手法より多くピースを拡散させていくことができた.多くピースを拡散することができたことによって,同じピースを所持しているピアがネットワーク内で多く存在することにより,一つのピアに要求が集中することを緩和することができ,待ち時間の低減や再送要求の減少を行うことができたと考えられる.

4.3.2　再生回数の増加における再生時間の変化

アドホックネットワーク環境では,サーバの位置によって再生までにかかる時間に大きな影響を受けると考える.これは,サーバの位置がストリーミングを行うピアから遠いほど,ホップ数が増加してしまうため,サーバに接続するまでに多く時間が掛かってしまうためである.サーバのホップ数の違いによって,ストリーミング再生にかかる時間が変化し,待ち時間や遅延などの問題点が発生する.そこで,提案手法では,サーバからすべてのピースを受信するのではなく,ピースを所有しているホップ数の最も近いピアからピースを受信することによって,時間短縮を図ることができると考える.評価方法としては,再生回数によりサーバ・クライアント方式と提案手法における再生時間を計測し,どのくらい再生時間を短縮することができるかを評価する.以下に,サーバからピースをすべて受信する場合と提案手法を用

いて受信する場合の再生回数による再生までにかかる時間を図 8 で表した.

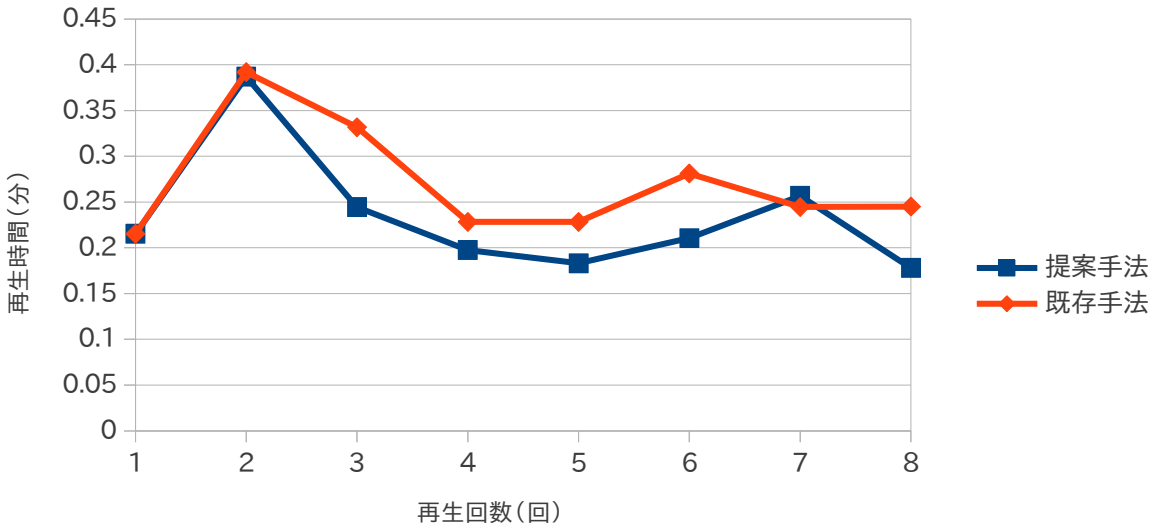

図 8, 再生回数における再生時間

図 8 より,再生回数が 2 回までは,提案手法と既存手法において再生までにかかる時間の差はなく,3 回目以降になると提案手法の方が,再生までにかかる時間を短く短縮することができている.これは,2 回目までは,再生回数がまだ少ないためストリーミング再生を行うピアの近くにピースを所有しているピアが存在していなかったため,既存手法と提案手法との再生時間に差が生じていないと考えられる.3 回目以降では,ある程度ネットワーク内でピースを拡散することができ,ストリーミング再生を行うピアの近くにピースを所有しているピアが存在する確率が大きくなったため,サーバからピースを取得するのではなく,ホップ数の少ないピアからピースを取得するようになりストリーミング再生にかかる時間が短縮できていると考えられる.

第５章　考察

本実験の結果より考察する.

5.1　ストリーミング再生におけるピースの総数

既存手法では,インターネット環境を用いてストリーミングを行っているため,中継ノードが存在せず,再生要求を行ったピアのみがピースを取得することによって（再生するピア数）*（コンテンツ再生までのピースの総数）個となり,また,既存手法+MANET では,中継ノードが存在するが,中継ノードは,ピースを経由するだけであり,記憶しないためピースの個数の増加に影響を与えないことから,既存手法と同じく,（再生するピア数）*（コンテンツ再生までのピースの総数）個となる.提案手法では,アドホックネットワーク環境でストリーミングを行っていくため,（再生するピア数）*（コンテンツ再生までのピースの総数）+（中継ノードが記憶したピース）個となり,再生回数が同じであっても,ネットワーク内でのピースの総数を既存手法や既存手法+MANET よりも多く拡散させていくことができたと考えられる.本来,アドホックネットワークの中継ノードは,ストリーミング再生を行うわけではないため,中継ノードが記憶するピースの個数を３個と設定し,実験を行ったが,このピースの記憶個数と記憶しておく時間を中継ノードに対して負担とならない値に最適化することによって,中継ノードの負荷を軽減させていくことができ,再生までにかかる時間を効率よく短縮することができると考える.

5.2　再生回数の増加に伴う再生にかかる時間

実験結果から,提案手法は,再生回数が３回目以降では,既存手法+MANET に比べ,提案手法が再生までにかかる時間を短縮することができた.これは,再生がすでに複数回行われていることにより,中継ノードにピースを記憶させていくことや他の再生が終了したピアが存在することによって,ネットワーク内でのピースの個数が増加したこと,また,サーバまでのホップ数より少ないホップ数のピアからピースを取得することによって,再生までにかかる時間が短縮されたと考える.これにより,サーバまでのホップ数が大きいピアであっても,再生回数が多くなることによって再生までにかかる時間を大きく短縮することができると考える.

5.3　考察のまとめ

5.2 節から,再生回数によってピースが拡散することで再生時間を短縮することができることが考えられる.これに,5.1 節の中継ノードのピースの記憶量の最適値を考慮することによって,中継ノードに負荷をかけず,ネットワーク全体の再生までにかかる時間をより効率よく短縮することができると考えられる.また,中継ノードにピースを記憶させていくことによって,希少性の低いピースをネットワーク内に早く拡散させていくことができ,再生時間の短縮を図ることができると考えられる.

第 6 章　終わりに

本研究は,インターネット環境で構築されている P2P ストリーミングをアドホックネットワーク環境で構築し,アドホックネットワークの特徴を利用したコンテンツピースの拡散と待ち時間の短縮を行う手法を提案した.再生回数が増加していくにつれ,提案手法の方が待ち時間の短縮とピースの効率のよい拡散を行うことができた.しかし,中継ノードにピースを記憶させることによって,ストリーミングを行っていないピアの記憶容量を使用することになり,負荷がかかってしまう問題点がある.中継ノードのピースの記憶容量の最大値を決めておくことによって,一定値までの負荷に保つことができる.

今後の課題として,中継ノードの記憶容量の負荷の最小限化を行うために,中継ノードのピースの記憶容量の最適値の計測と時間の経過によってピースを消去していくことなどのストリーミングを行っていないピアの負荷の低減を行っていくことを進める.また,中継ノードが希少値の高いピースを記憶していくことによって,BIS 方式などの希少値に大きく影響を与えると考えられる.これより,重要度の計算を中継ノードがピースを記憶してくことを考慮した上で,重み定数 c の測定を行い,アドホックネットワーク環境に沿った重要度の計算式を作成していく.

謝辞

本研究,論文を行なっていくに当たり,多くの御指導と御助言をいただきました三好力教授に感謝いたします.また,多くの御助言をいただきました三好研究室の皆様に感謝いたします.

参考文献

[1]「P2P ストリーミング環境における再生の途切れ時間短縮のための分割データ受信方式」
<http://www-nishio.ist.osaka-u.ac.jp/Thesis/bachelor/2011/yokoyama/thesis.pdf>

[2]「ストリーミングとは」
<http://www.realstream.jp/streaming/>

[3]「ユビキタス社会を目指して現実化するアドホック・ネットワーク」
<http://wbb.forum.impressrd.jp/feature/20071210/514>

[4] 銭飛著「実験で学ぶ QoS ネットワーク技術 NS2 によるネットワークシミュレーション」

[5] 水野秀樹著「NS2 によるネットワークシミュレーション入門 有線からワイヤレスアドホックネットワークまで」

付録

```
Mac/Simple set bandwidth_ 1Mb

set MESSAGE_PORT 42
set BROADCAST_ADDR -1


# variables which control the number of nodes and how
they're grouped
# (see topology creation code below)
set group_size 7
set num_groups 7
set num_nodes [expr $group_size * $num_groups]

set val(chan)           Channel/WirelessChannel
;#Channel Type
set val(prop)           Propagation/TwoRayGround   ;# 距離
が大きく大地に反射されることを考慮した大地反射モデルの設定
set val(netif)          Phy/WirelessPhy            ;# network
interface type

#set val(mac)           Mac/802_11                 ;# MAC type
#set val(mac)           Mac                ;# MAC type
set val(mac)                    Mac/Simple

set val(ifq)            Queue/DropTail/PriQueue    ;#
interface queue type
set val(ll)             LL                         ;# link layer type
set val(ant)            Antenna/OmniAntenna        ;# antenna
model
set val(ifqlen)         50                         ;# max packet in ifq

# DumbAgent, AODV, and DSDV work.  DSR is broken
set val(rp) DumbAgent
#set val(rp)            DSDV
#set val(rp)            DSR
#set val(rp)                    AODV

# size of the topography
set val(x)              1500;#[expr 120*$group_size + 500]
set val(y)              1500;#[expr 240*$num_groups + 200]

for {set i 0} {$i <= $num_nodes} {incr i} {
   set cliant($i) "NO"
}

set sarver 24
set cliant2(1) 19
set cliant2(2) 27
set cliant2(3) 29
set cliant2(4) 11
set cliant2(5) 46
set cliant2(6) 7
set cliant2(7) 23
set cliant2(8) 2

set p_count 0
for {set i 0} {$i <= 8} {incr i} {
   set p_count_box($i) 0
}

set p_contral 0

set p_k 0

set ka 8

for {set i 1} {$i <= $ka} {incr i} {
   set cliant($cliant2($i)) "ON"
}

#set side 300
#set higt 200

set peace_count 1

for {set i 0} {$i < $num_nodes} {incr i} {
   set hop($i) 0
}

for {set i 0} {$i < $num_nodes} {incr i} {
   for {set j 1} {$j < 11} {incr j} {
            if {$i != $sarver} {
               set g 0
            } else {
               set g 1
            }
            set a($i,$j) "$g"
   }
}

for {set i 0} {$i < $num_nodes} {incr i} {
   for {set j 1} {$j < 11} {incr j} {
            set b($i,$j) "NO"
            #            puts "¥n b($i,$j) : $b($i,$j) ¥n"
   }
}

set sender 0

for {set j 0} {$j < $num_nodes} {incr j} {
   for {set i 1} {$i < 20} {incr i} {
            set root_sample($j,$i) "0"
   }
}

set count2 1
```

```
for {set i 1} {$i < 11} {incr i} {
    set peace($i) $i
    #    puts "¥n peace($i) : $peace($i) ¥n"
}

for {set i 1} {$i < 11} {incr i} {
    set peace_get($i) 0
    #    puts "¥n peace_get($i) : $peace_get($i) ¥n"
}

for {set i 0} {$i < $num_nodes} {incr i} {
    set peace_get_count($i) 0
}

set ns [new Simulator]

set f [open test.tr w]
$ns trace-all $f
set nf [open wireless-flooding-$val(rp).nam w]
$ns namtrace-all-wireless $nf $val(x) $val(y)

$ns use-newtrace

# set up topography object
set topo       [new Topography]

$topo load_flatgrid $val(x) $val(y)

#
# Create God
#
create-god $num_nodes

set chan_1_ [new $val(chan)]

$ns node-config -adhocRouting $val(rp) ¥
    -llType $val(ll) ¥
    -macType $val(mac) ¥
    -ifqType $val(ifq) ¥
    -ifqLen $val(ifqlen) ¥
    -antType $val(ant) ¥
    -propType $val(prop) ¥
    -phyType $val(netif) ¥
    -topoInstance $topo ¥
    -agentTrace ON ¥
    -routerTrace OFF ¥
    -macTrace ON ¥
    -movementTrace OFF ¥
    -channel $chan_1_

# subclass Agent/MessagePassing to make it do flooding

set look "NO"
for {set i 0} {$i < $num_nodes} {incr i} {
    set k_k($i) 0
    set select_peace($i) 0
}
```

```
Class Agent/MessagePassing/Flooding -superclass
Agent/MessagePassing

Agent/MessagePassing/Flooding instproc recv {source
sport size data} {
    $self instvar messages_seen node_
    global ns BROADCAST_ADDR num_nodes look
peace_get_count
    global hop a b root_sample peace peace_get
peace_count sender k_k select_peace
    global cliant sarver p_count p_count_box p_contral p_k

    # extract message ID from message
    set message_id "[lindex [split $data ":"] 0]:[lindex [split
$data ":"] 1]" ;#dateを:によって分割後、抽出した値を
message_idに代入
    set message_id_sarvar [lindex [split $data ":"] 0]
    set message_log [lindex [split $data ":"] 2]
    set count [lindex [split $data ":"] 3]
#    puts "¥nNode [$node_ node-addr] got message
$message_id and $sport and $size and $source and $data
and $look and $messages_seen¥n"

    if {[lsearch $messages_seen $message_id] == -1 &&
$message_id_sarvar != [$node_ node-addr] &&
$message_log == "first message" && $look == "NO"} {
;#messages_seenにmessage_idがあるかを確認しない場合 if
の動作を行う

#            puts "¥nNode [$node_ node-addr] got message
$message_id and $sport and $size and $source and
$data¥n"
            lappend messages_seen $message_id
;#massages_seenにIDを追加している

            for {set i 1} {$i < 11} {incr i} {
                if {$a([$node_ node-addr],$i) == 1} {
                        set b([$node_ node-addr],$i)
$a([$node_ node-addr],$i)
                }
                #            puts "¥n jouhou : $b([$node_
node-addr],$i) ¥n"
            }

            if {$count == 1} {
                set root_sample([$node_ node-addr],$count)
$source
            } else {
                for {set i 1} {$i < $count} {incr i} {
                        set root_sample([$node_ node-addr],
$i) $root_sample($source,$i)
                }
                set root_sample([$node_ node-addr],$count)
$source
            }

            set hop([$node_ node-addr]) $count;#カウントの
増加とホップ数の確保
```

```
            incr count

            set root_sample([$node_ node-addr],$count)
[$node_ node-addr]

            $ns trace-annotate "[$node_ node-addr]
received {$data} from $source"
            $ns trace-annotate "[$node_ node-addr] sending
message $message_id"

            incr p_count
            $self sendto $size "$message_id:first message:
$count" $BROADCAST_ADDR $sport

    } else {

            if {[$node_ node-addr] == $message_id_sarvar
&& $message_log == "first message"} {

                for {set i 1} {$i < 11} {incr i} {
                        set b([$node_ node-addr],$i)
$a([$node_ node-addr],$i)
                        #                       puts "¥n [$node_
node-addr] : $b([$node_ node-addr],$i) ¥n"
                }

                set look "OK"

                if {$count == 1} {
                        set root_sample([$node_ node-addr],
$count) $source
                } else {
                        for {set i 1} {$i < $count} {incr i} {
                           set root_sample([$node_ node-addr],
$i) $root_sample($source,$i)
                        }
                        set root_sample([$node_ node-addr],
$count) $source
                }

                set hop([$node_ node-addr]) $count;#カウント
の増加とホップ数の確保
                incr count

                set root_sample([$node_ node-addr],$count)
[$node_ node-addr]

                for {set i 1} {$i <= $count} {incr i} {
                        puts "¥n root_sample([$node_
node-addr],$i) : $root_sample([$node_ node-addr],$i) ¥n"
                }

                incr p_count
                $self sendto $size "OK:OK:second message:
$count" $source $sport
            }

            if {$message_log == "second message"} {

                if {[$node_ node-addr] == $sender} {
                        incr p_count
                        $self packet_send $size $source "third
message" $sport
                } else {
                        incr p_count
                        $self sendto $size "OK:OK:second
message:$count" $root_sample([$node_ node-addr],
$hop([$node_ node-addr])) $sport
                }
            }

            if {$message_log == "third message"} {

                if {[$node_ node-addr] ==
$message_id_sarvar} {
                        incr p_count
                        $self packet_send_return $size $sport
$source $message_log
                } else {
                        if {$p_k == 0 && $source == $sender} {
                           set p_count_box([expr
$p_contral+1]) $p_count
                           set p_count 0
                           incr p_k
                           incr p_contral
                        }
                        set sample $hop([$node_ node-addr])
                        incr sample
                        incr sample
                        incr p_count
                        $self sendto $size $data
$root_sample($message_id_sarvar,$sample) $sport
                }
            }

            if {$message_log == "forth message"} {

                if {[$node_ node-addr] == $sender} {
                        set peace_get($peace_count)
$message_id_sarvar
                        puts "¥n [$node_ node-addr] peace get
$peace_get($peace_count) ¥n"
                        $ns trace-annotate "[$node_
node-addr] get peace $peace_get($peace_count)"

                        set b([$node_ node-addr],
$peace_get($peace_count)) 1

                        if {$peace_get($peace_count) != 10} {
                           incr p_count
                           $self packet_send $size $source
$message_log $sport
                        } else {
                           puts "¥n end ¥n"

                           for {set i 0} {$i < $num_nodes} {incr
i} {
                                   for {set j 1} {$j < 11} {incr j} {
```

```
                        if {$b($i,$j) == 1} {
                                set a($i,$j) $b($i,$j)
                        }
                    }
                }

                for {set i 0} {$i < $num_nodes} {incr
i} {
                    for {set j 1} {$j < 11} {incr j} {
                        if {$sarver != $i && $a($i,
$j) == 1 && $cliant($i) == "NO"} {
                                puts "¥n a($i,$j) =
$a($i,$j) ¥n"
                        }
                    }
                }

                set look "NO"
                set peace_count 1

                for {set i 0} {$i < $num_nodes} {incr
i} {
                    set peace_get_count([$node_
node-addr]) 0
                }

                set fin 0

                for {set i 0} {$i < $num_nodes} {incr
i} {
                    for {set j 1} {$j < 11} {incr j} {
                        if {$a($i,$j) == 1} {
                                incr fin
                        }
                    }
                }

                for {set i 0} {$i < $num_nodes} {incr
i} {
                    for {set j 1} {$j < 11} {incr j} {
                        set b($i,$j) "NO"
                        set k_k($i) 0
                    }
                }

#               set p_count_box([expr
$p_contral+1]) $p_count
                puts "¥n finish : peace is $fin ¥n"
                set p_count 0
                set p_k 0
#               incr p_contral
                if {$p_contral == 8} {
                    for {set p_i 1} {$p_i <= 8}
{incr p_i} {
                        puts "p_count is
$p_count_box($p_i)"
                    }
                }
            }
        } else {
            puts "¥n streaming now return [$node_
node-addr]¥n"
            set paket_input [lindex [split $data ":"]
1]

            set k_l_2 1000

            if {$k_k([$node_ node-addr]) == 0} {
                for {set j 1} {$j < 11} {incr j} {
                    set k_l 0
                    for {set i 0} {$i <
$num_nodes} {incr i} {

                        if {$a($i,$j) == 1} {
                            incr k_l
                        }

                    }

                    if {$k_l < $k_l_2} {
                        set k_l_2 $k_l
                        set select_peace([$node_
node-addr]) $j

                    }
                    puts "¥n [$node_ node-addr]
is $select_peace([$node_ node-addr]) sum $k_l $k_l_2¥n"
                }
                set k_k([$node_ node-addr]) 1
            }

            if {$peace_get_count([$node_
node-addr]) < 3 && $b([$node_ node-addr],
$message_id_sarvar) == "NO" && $message_id_sarvar ==
$select_peace([$node_ node-addr])} {
                set b([$node_ node-addr],
$message_id_sarvar) 1
                set a([$node_ node-addr],
$message_id_sarvar) $b([$node_ node-addr],
$message_id_sarvar)
                incr peace_get_count([$node_
node-addr])
                incr p_count
                $self sendto $size
"$message_id_sarvar:NO:$message_log:$count"
$root_sample([$node_ node-addr],$hop([$node_
node-addr])) $sport
            } else {
                incr p_count
                $self sendto $size $data
$root_sample([$node_ node-addr],$hop([$node_
node-addr])) $sport
            }
        }
    }
```

```
}

Agent/MessagePassing/Flooding instproc
packet_send_return {size port source message_log} {
   $self instvar message_seen node_
   global ns MESSAGE_PORT BROADCAST_ADDR hop a
b root_sample peace peace_get peace_count p_count

   if {$a([$node_ node-addr],$peace_count) == "1"} {
            puts "¥n OK ¥n"
            incr p_count
            $self sendto $size
"$peace($peace_count):NO:forth message:$peace_count"
$source $port
            $ns trace-annotate "[$node_ node-addr] sending
peace $peace($peace_count)"
   } else {
            puts "¥n Noooooooooooo ¥n"
   }

}

Agent/MessagePassing/Flooding instproc packet_send
{size source message_log port} {
   $self instvar messages_seen node_
   global ns MESSAGE_PORT BROADCAST_ADDR
num_nodes
   global hop a b root_sample peace peace_count sarver
p_count
   #    lappend messages_seen $message_id
   #    $ns trace-annotate "[$node_ node-addr] sending
message $message_id"

   set select 1

   if {$message_log == "forth message"} {
            incr peace_count
   }

   if {$b([$node_ node-addr],$peace_count) == 1} {
            incr $peace_count
   }

   set teach "NO"

   for {set i 0} {$i < $num_nodes} {incr i} {
            if {$b($i,$peace_count) == 1} {
               if {$hop($i) < $hop($sarver) && $teach ==
"NO"} {
                        set teach $i
                        set select 0
               } else {
                        if {$teach != "NO" && $hop($i) <
$hop($teach)} {
                           set teach $i
                        }
               }
            }
   }

   puts "¥n $teach and $select and [$node_
node-addr] ¥n"

   puts "¥n Streaming start get peace
$peace($peace_count) ¥n"
   $ns trace-annotate "[$node_ node-addr] sending
peace_message $peace_count to get peace"
   puts "¥n size:$size source:$source message_log:
$message_log port:$port ¥n"

   if {$select == 1} {
            set message_id_sarvar $sarver
            set message_id "$sarver:[$node_ node-addr]"
            incr p_count
            $self sendto $size "$message_id:third message:
$peace_count" $root_sample($message_id_sarvar,1)
$port
   } else {
            set message_id_sarvar $teach
            set message_id "$teach:[$node_ node-addr]"
            incr p_count
            $self sendto $size "$message_id:third message:
$peace_count" $root_sample($message_id_sarvar,1)
$port
   }

}

Agent/MessagePassing/Flooding instproc send_message
{size message_id data port} {
   $self instvar messages_seen node_
   global ns MESSAGE_PORT BROADCAST_ADDR
sender p_count
   set messages_seen {}

   lappend messages_seen $message_id
   $ns trace-annotate "[$node_ node-addr] sending
message $message_id"
   set sender [$node_ node-addr]
   puts "¥n message_id:$message_id ¥n"
   set count2 1
   incr p_count
   $self sendto $size "$message_id:$data:$count2"
$BROADCAST_ADDR $port

}

set group 0
set k 0

for {set i 0} {$i < $num_nodes} {incr i} {
   set n($i) [$ns node]
   if {$k == 0} {
            $n($i) set X_ 0
            $n($i) set Y_ [expr $group*250]
            $n($i) set Z_ 0
```

```
            $ns initial_node_pos $n($i) 20
    } else {
            if {$k < $group_size} {
                $n($i) set X_ [expr $k*250]
                $n($i) set Y_ [expr $group*250]
                $n($i) set Z_ 0
                $ns initial_node_pos $n($i) 20
            }
    }

    set k [expr $k+1]

    if {$k == $group_size} {
            set k 0
            set group [expr $group+1]
    }
}

# attach a new Agent/MessagePassing/Flooding to each
node on port $MESSAGE_PORT
for {set i 0} {$i < $num_nodes} {incr i} {
    set q($i) [new Agent/MessagePassing/Flooding]
    $n($i) attach  $q($i) $MESSAGE_PORT
    $q($i) set messages_seen {}
}

# now set up some events

$ns at 0.1 "$q($cliant2(1)) send_message 200 {$sarver:
$cliant2(1)} {first message}  $MESSAGE_PORT"
$ns at 1.0 "$q($cliant2(2)) send_message 200 {$sarver:
$cliant2(2)} {first message}  $MESSAGE_PORT"
$ns at 2.0 "$q($cliant2(3)) send_message 200 {$sarver:
$cliant2(3)} {first message}  $MESSAGE_PORT"
$ns at 3.0 "$q($cliant2(4)) send_message 200 {$sarver:
$cliant2(4)} {first message}  $MESSAGE_PORT"
$ns at 4.0 "$q($cliant2(5)) send_message 200 {$sarver:
$cliant2(5)} {first message}  $MESSAGE_PORT"
$ns at 5.0 "$q($cliant2(6)) send_message 200 {$sarver:
$cliant2(6)} {first message}  $MESSAGE_PORT"
$ns at 6.0 "$q($cliant2(7)) send_message 200 {$sarver:
$cliant2(7)} {first message}  $MESSAGE_PORT"
$ns at 7.0 "$q($cliant2(8)) send_message 200 {$sarver:
$cliant2(8)} {first message}  $MESSAGE_PORT"

#$ns at 0.4 "$a([expr $num_nodes/2]) send_message 600
2 {some big message} $MESSAGE_PORT"
#$ns at 0.7 "$a([expr $num_nodes-2]) send_message 200
3 {another one} $MESSAGE_PORT"

$ns at 20.0 "finish"

proc finish {} {
    global ns f nf val
    $ns flush-trace
    close $f
    close $nf

    #       puts "running nam..."
#    exec nam wireless-flooding-$val(rp).nam &
    exit 0
}

$ns run
```