卒業論文　　2004年度 (平成16年度)

ネットワークゲームの再利用を可能にする通信接続形態切替機構

指導教員

慶應義塾大学 環境情報学部

徳田　英幸

村井　純

楠本　博之

中村　修

南　政樹

慶應義塾大学 環境情報学部

金田 裕剛

卒業論文要旨　　2004 年度 (平成 16 年度)

ネットワークゲームの再利用を可能にする通信接続形態切替機構

常時接続回線の急速な浸透からネットワークを介して大人数で遊ぶネットワークゲームが普及してきている。ネットワークゲームの通信接続形態には、主に中央ゲームサーバを利用する C/S(クライアント・サーバ) 型、ユーザ同士を直接接続する P2P(ピア・ツー・ピア) 型の 2 種類が存在する。現在運用されている商用のネットワークゲームはコンテンツ管理の容易さから大半が C/S 型を利用する。商用ネットワークゲームでは時間推移とともに参加ユーザ数が減少しゲームサーバ管理費等のサービス維持費用を確保できなくなり、サービス停止に至るゲームが増加傾向にある。一方で流行が終わり一度は廃れてしまった懐かしいコンシューマゲームを再販し利益を上げている例も多く存在し、エンタテインメントとしてのゲームは再利用性という利点を持っている。ネットワークゲームにおいて再利用を実現する場合、コンシューマゲームと違いネットワークを介してサーバ資源に接続してサービスを享受するという特性上、新たにサービスを提供する企業又は個人が必要となる。だが採算の保障がないサービスを再提供する第 3 者の出現を期待することは現実的に難しい。理想的な再利用の一形態としてはサービス提供側はソフトウェアのみを販売し、ゲームを遊ぶユーザ自身が C/S 型通信接続形態のソフトウェアをサーバに依存しない自律的なゲームネットワークを構築可能な P2P 型通信接続形態に切り替えて遊べることが望ましい。C/S 型から P2P 型へ通信接続形態の切り替えを実現するにはソフトウェア自体の通信部分を書き換えて P2P 型に変更するコストが必要となる。

本研究では、まずネットワークゲームの再利用時における問題を社会的側面、技術的側面から考察し、ネットワークゲームの再利用を可能にするための前提条件を定義する。そしてネットワークゲームの再利用を可能にするため、C/S 型および P2P 型ネットワークの切り替えを既存ソフトウェアの変更なしで実現する機構を考察に基づき設計し実装する。また既存ソフトウェアへの対応のみならず、将来的に新しいネットワークゲームを開発する際に利用できるフレームワークとしての開発環境も提案する。最後に本機構を実機にて評価する。

P2P 型通信ネットワークゲームにおいては一般的にメッシュ型ネットワークを構築するが、この手法ではネットワークに参加するノードが増えるにつれ、一部通信経路にて遅延が発生する可能性がある。そこで本研究のトポロジ構築手法においてネットワークゲーム向けのメッシュ型アプリケーションレベルマルチキャストの最適化機構を提案し、試作実装を使用し簡単な性能評価も行う。

慶應義塾大学 環境情報学部
金田 裕剛

Abstract of Bachelor's Thesis
A Generic Framework for Switching Network Architecture to Reuse Multiplayer Online Games

Multiplayer Online Gaming (MOG) came into wide use along with the popularization of various broadband networks and the technical advancement of gaming devices. Quake [15], Age of the Empire [20], and Counter-Strike [35] are the examples of the most popular game titles. In these network games, we can share the same virtual gaming space and communicate to play the game with other players who join the same game at the same time through the Internet.

When we refer to the network architecture of online multiplayer games, it is divided into two main categories which are Client-Server architecture (C/S) and Peer-to-Peer architecture (P2P). In C/S, every player who wants to share the same gaming space connects to a game server which calculates the player's interaction and transition of the whole game state, and players update their game states based on the results which are sent by the game server. Most commercial online games adopt C/S architecture. It is clear that game management is easy for game providers because they can manage gaming environments centrally and distribute update programs to all users at once. However, players cannot use gaming service when the game server is down.

Recently, there is a serious problem that it is difficult to continue hosting a gaming service when the game becomes less popular. The reason is that game providers must continue paying for maintaining game servers. Additionally, it is not clear whether there are any organizations or any individuals who would support to continue providing such game services for the future. Thus, users which play such a unpopular network games are forced to stop enjoying the game services.

On the other hand, P2P has no central management node for handling players and calculation of game states. This architecture forms distributed network by connecting players directly and has no single point of failure such as game server.

There are many successful commercial cases to resell old games at a low price. We define this as "reuse of MOG."

It is difficult to resell such commercial MOGs because we must consider the cost to mantain game servers. Thus it is efficient for game providers to be supplied with reusing architecture for MOG because they can increase the chance to increase sales for such games.

One of the solutions for realizing reuse of MOG is to exchange network architecture of such games from C/S to P2P for the reason that we do not need the cost to maintain game servers. At the same time, when we realize it, we must reprogram network parts of its game software. However, for general users, it is overly complicated.

In this thesis, we propose a generic framework for reusing Multiplayer Online Games.

This framework makes it possible for game providers and general users to change network architecture from C/S to P2P without reprogramming game software.

Furthermore, MOGs which adopt P2P architecture construct mesh network in general. However, it increases possibility to include high delay paths among users. We propose an optimization method for application-level multicast in MOG, and have implemented its prototype. We have done the simple evaluation in this thesis.

Yugo Kaneda

Faculty of Environmental Information
Keio University

目 次

図 目 次

表 目 次

第1章　序論

1.1　研究動機

近年常時接続回線の急速な浸透からネットワークを介して多人数で遊ぶネットワークゲームが普及してきている。大規模ネットワーク仮想空間 (VR) にて本格的な軍事シミュレーションを実現する研究から発展した本分野は現在までに様々なソフトウェアが登場し、世界中の人々がネットワークに接続し、ゲームを楽しんでいる [28]。有名なものとして Quake [15] や Age of the Empire [20], Counter Strike [35] が挙げられる。また従来家庭用ゲームであるコンシューマゲームの開発をしていた大手企業もネットワークゲームを積極的に発売しており、日本で上位の売上を記録しているファイナルファンタジーシリーズのオンライン版であるファイナルファンタジー XI [32] も人気を博している。

昨今のネットワークゲームはクライアント・サーバ型 (C/S) という形式を取る。この形式はゲーム共有計算資源であるユーザ間のデータ管理や、操作の計算と結果の反映などをサーバ側に構築しブラックボックス化する。そしてクライアントであるユーザにはグラフィックス処理やサウンド処理などマルチメディア計算処理が行われる。サーバが各ユーザに更新メッセージを送信する時間やデータベースで一括してユーザ履歴データを管理できるため、ユーザ間で公平でセキュアなゲーム展開を保証できる。しかしゲームサーバがサービス稼動において必須となるため、ゲームサーバを管理する人材が常に必要となり、自ずとサービス維持費用がユーザに請求される。これはネットワークゲームを全く遊んでいないユーザにおいてもサービス料金を支払い続けなければならない状況が発生する。このため人気のなくなってきたネットワークゲームからはユーザが早急に契約解除をする傾向にある。人気がなくなり、サービス維持管理費用をユーザから十分収集できなくなったことでサービス廃止に追い込まれるネットワークゲームが存在する。そのようなネットワークゲームで常時遊んでいたユーザは強制的に終了を迫られてしまう。

だが一方で過去のゲームが復刻版として低価格で再販され著しい売上げを収める場合が近年見られる。ソニーコンピュータエンタテインメント [30] が行っている PlayStation The Best [29] シリーズや任天堂 [24] のファミコンミニシリーズ [23] が例として挙げられる。これは音楽や映画同様、ゲームがエンタテイメント性を持つ商品であるため、時と場合により改めて面白さや長所をユーザから再認識される場合があるからである。ゲーム提供側は莫大な開発費を投資して各ゲームソフトウェアを発売するため、商業的に成功する機会を増やす方法としてゲームの再販売は有用性が高い。本論文ではこのようにゲームを再販売することを再利用と定義する。ネットワークゲームも従来の

コンピュータゲーム同様莫大な開発費をかけて開発されている。商業的に成功する機会を増やすためにネットワークゲームを再利用できる環境が必要である。

ネットワークゲームの再利用を考慮した場合、前述したようにサーバ管理費用を必ず考慮しなければならない。しかし商業的に成功するかどうか不透明な場合、サーバを維持するサービス提供者が常に現れるとも限らない。そこで再利用におけるサーバ管理コストを削減またはなくす理想的な手段として、各ユーザにクライアント資源とサーバ資源を包括させ、ユーザ同士で直接接続を行い自律的にゲームを展開する P2P 型ネットワークで遊ばせることが考えられる。

だがここで新たな問題が存在する。それは既存の商用ネットワークゲームが C/S 型通信を採用しているため、P2P 型通信に切り替えるためにソフトウェア自体の通信を実装し直さなければならないことである。ゲームを再実装する場合、実装するための人材や開発環境の整備等多額の費用が必要となる。

ネットワークゲームの接続形態を C/S 型から P2P 型へ切り替えるもう一つの方法としてユーザにソースを公開し、実装を委任する場合が考えられる。しかしこの方法はユーザにとって非常に敷居が高い上、実現できる可能性も低くなる。またゲームごとに実装し直さなければならないため多数のゲームを P2P 型通信形態に変えることは非現実的である。

したがってネットワークゲームを再利用する基盤としてソフトウェアの変更なしで、通信形態の切り替えを容易にできる機構が必要となる。

1.2 本研究の目的及び意義

本研究ではネットワークゲームにおける通信形態切替機構の提案を行う。前述したように従来ではネットワークゲームの再利用を実現する場合に通信形態部分を実装し直さなければならなかったが、本機構はゲームソフトウェアとは独立した通信形態切替基盤を提供し、サービス提供側及びユーザ自身による簡易な通信形態切り替えを可能にする。

また本研究の設計の際には、今後もネットワークゲームが発展することを考慮し、将来的な開発フレームワークについても設計に含め言及する。

1.3 本論文の構成

本論文は全 6 章から構成される．2 章で研究背景であるネットワークゲームの概要と一般的な通信基盤技術を説明し、ネットワークゲーム全般における問題に触れた後、本論文で取り上げるネットワークゲーム再利用時の問題について言及する。そして 3 章でネットワークゲーム再利用を既存ソフトウェアの改変なく実現する通信接続形態切替機構の設計を説明する。次の 4 章では前章の設計に基づき行った本機構の実装の詳細について説明する。5 章にて本システムの有用性を評価にて実証し、最後に 6 章において本機構の今後の課題とまとめについて述べる。

第2章　研究背景と問題定義

本章では研究の背景と本研究で取り上げる問題について説明をする。

2.1　ネットワークゲーム概要

本節ではネットワークゲーム全体の概要を述べた後、コンテンツ的側面や通信基盤的側面に分類して述べる。

従来、コンピュータゲームは室内のようなごく物理的に狭い範囲で遊ぶことを前提に設計されており、単一機器内で全て処理されていた。そのため同じ仮想空間を通して同時にゲームを遊ぶ相手は常に顔が見える位置に存在する人に限定されていた。しかし常時接続回線の普及から様々なネットワークを利用したアプリケーションが登場し、同時にコンピュータゲームもネットワークを介したメッセージ通信を利用することでゲーム空間を共有するネットワークゲームが登場した。ネットワーク化によってゲームで遊ぶ相手が近距離内で存在しなければならないという前提は除去され、ネットワークを通して世界中で同じゲームを遊ぶ人々と一緒に仮想空間の共有が可能になった。

ネットワークゲームのユーザ数は増大し続けている。商用において大規模な参加人数で遊ぶネットワークゲームを示す Massively Multiplayer Online Game (MMOG) の 1997 年から 2004 年にかけての主要な人気ネットワークゲームにおけるユーザ数を図 2.1 に示す [6]。本資料は正確な計測値ではなく、大凡の数ではあるが、既に 275 万人のユーザが確認されている。また本資料には MMOG 以外で商用サーバを利用しないネットワークゲームのユーザ数は含まれておらず、実際のユーザ数はより膨大になる。今後も多種に渡るネットワークゲームが登場し市場は発展していくのが明確に予測できる。

2.1.1　ゲームジャンルの種類

現在様々なネットワークゲームが存在しているが、ゲームのジャンルには大きく分けて 3 種類存在する。それは First Person Shooting (FPS)、Real Time Strategy (RTS)、Massively Multiplayer Online Role Playing Game (MMORPG) である。それぞれの概要と特徴について簡単に触れる。

- First Person Shooting (FPS)

  FPS は一人称による主観視点でユーザのアバタとなるゲーム空間のキャラクタを操作し、敵 AI (Artificial Intelligence) のアバタ又は他のユーザと銃撃戦を繰

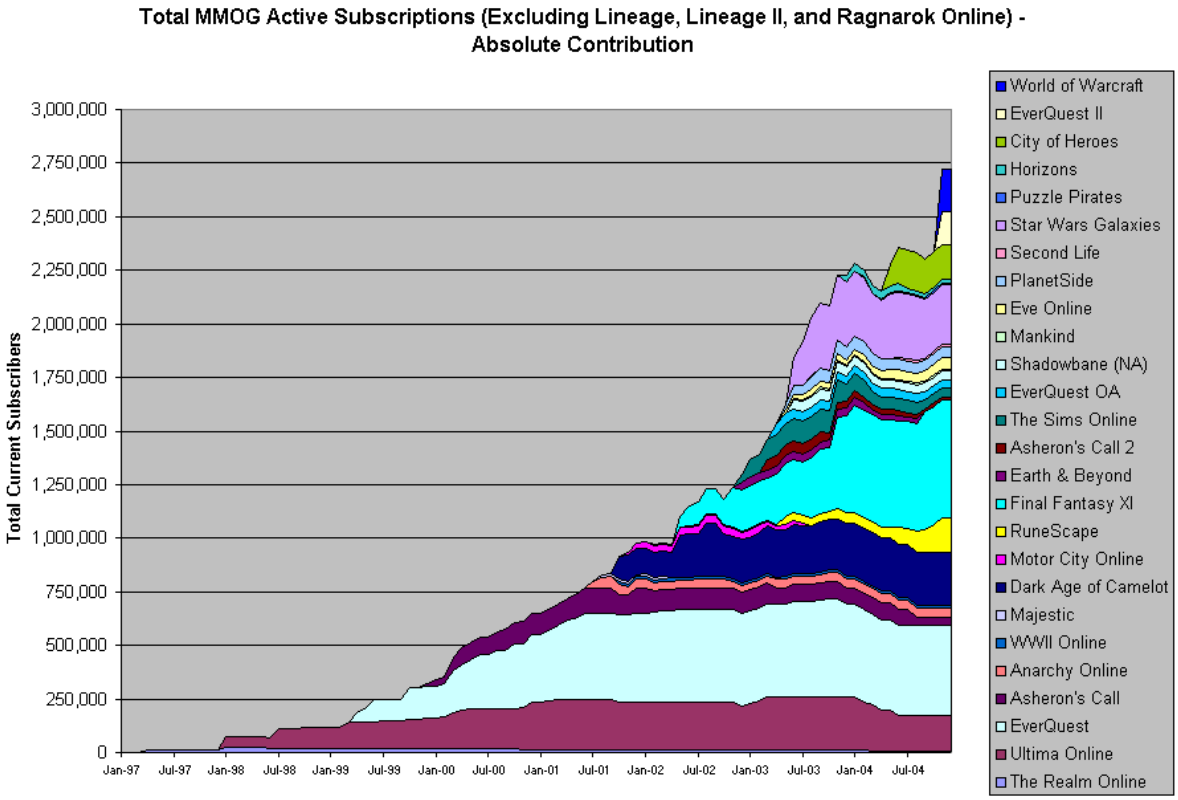

図 2.1: 主要ネットワークゲームユーザ数 (出典:MMOGCHART [6])

り広げ対戦をするゲームである。代表的なものとして Quake、Counter Strike、HALO [21] が挙げられる。

- Real Time Strategy (RTS)

  RTS はユーザがゲーム内における一国の主となりいくつかのプレイヤーが同時に参加し戦略を立ててお互いの領域を攻め合い戦うゲームである。代表的なゲームとして WarCraft [4] や Age of the Empire [20] が挙げられる。

- Massively Multiplayer Online Role Playing Game (MMORPG)

  MMORPG はロールプレイングゲームと呼ばれるゲーム空間上に壮大な物語の舞台と詳細な世界観や文化を構築し、その世界を旅行しているような気分で物語の進行を進めるゲームである。代表的なものとしてはラグナロクオンラインやファイナルファンタジー XI などが挙げられる。参加するプレイヤの数はゲームによって規模が変わるが、最大数十万人規模のゲームも存在する。

2.1.2 通信基盤機構

ネットワークゲームで利用される通信基盤の技術は大きく分けて 2 つに分類される。一つはクライアント・サーバ (C/S) 型ともう一つはピア・ツー・ピア (P2P) 型通信形態である。

クライアント・サーバ (C/S) 型

C/S 型通信では、ネットワークゲームを構成する端末の役割を主に 2 種類に分ける。ゲームに参加するユーザをクライアント、ゲーム空間での位置計算といった全ユーザが共有するデータ管理を行う端末をサーバとする。各クライアントはゲーム参加時には必ずサーバへ接続を行う。クライアントからはゲーム空間に対して操作した動作がユーザ操作メッセージとしてサーバへ送られ、その動作を反映した結果がサーバから各クライアントへ返信される。クライアントとサーバの更新を繰り返すことでゲームが展開される。

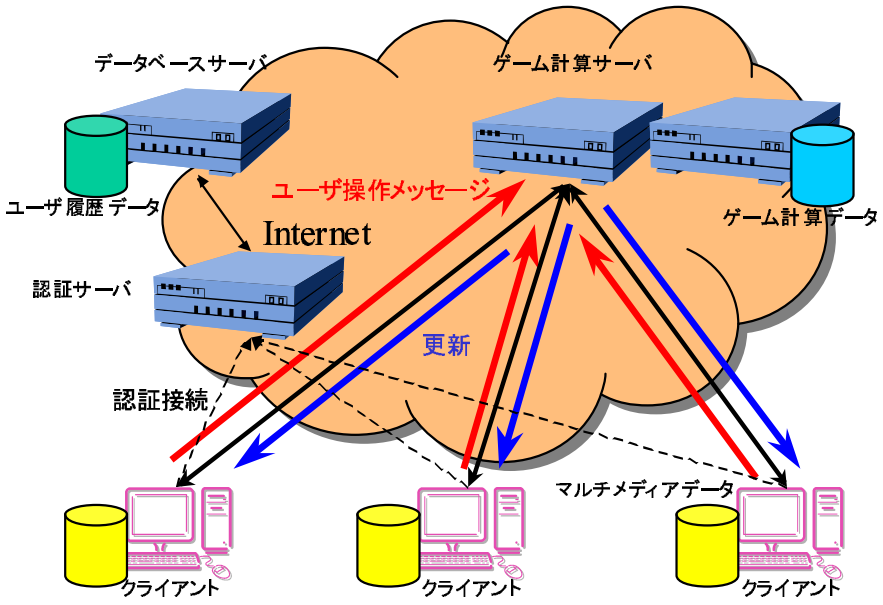

図 2.2: C/S 型汎用通信設備環境

商用 C/S 型ネットワークゲームにおける通信トポロジの汎用例を図 2.2 に挙げる。クライアント側は大抵ユーザ一人あたりに一台の端末でゲームに参加する。サーバ側

は大量のクライアントからのメッセージ処理を行うために、機能ごとに構成の負荷を分散させる場合が多い。基本構成として正当なユーザであるかを検証する認証管理サーバ、認証管理のためのユーザ履歴データを蓄積するデータベースサーバ、ゲーム空間処理を行う計算サーバで構成される。

C/S 型通信では、ゲームに参加するユーザはゲーム空間に大して何らかの操作をするインタフェイスを保持する。サーバではユーザの操作内容に応じたメッセージを反映し、ユーザ同士のゲーム空間上での位置やイベントの変化を計算処理し結果をクライアントに返す。

ピア・ツー・ピア (P2P) 型

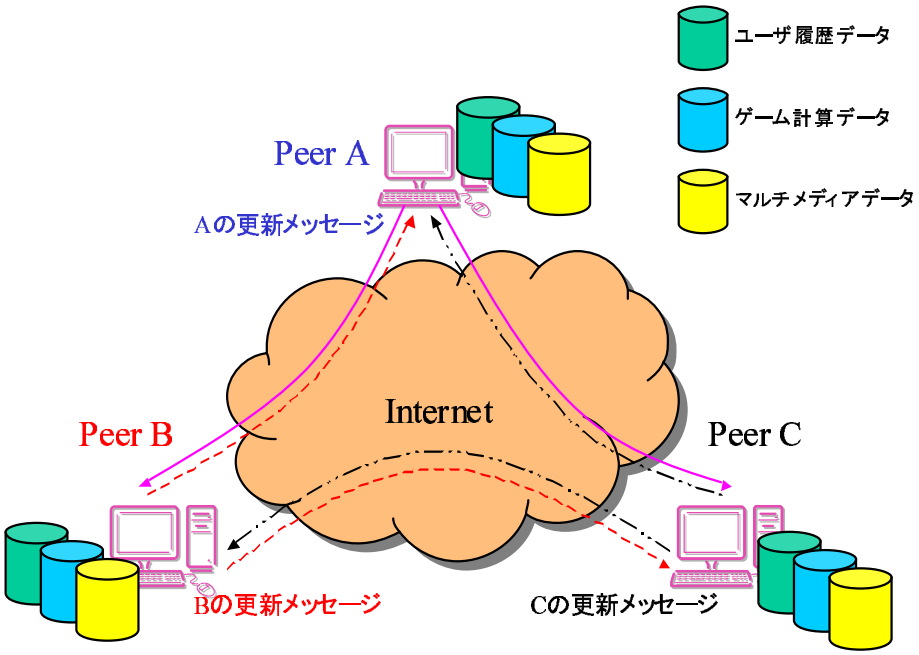

図 2.3: P2P トポロジ概要図

C/S 型と対照的な接続形態として P2P 型通信が挙げられる。P2P 型ネットワークトポロジの例を図 2.3 に示す。P2P 型では各ユーザの端末がネットワークを介して平等な関係に接続されることで分散型トポロジとなる。そのため中央で管理を行うサーバが存在せず、負荷が分散されることで単一故障点がない利点がある。ただしサービスの一元管理が不可能となり、異種通信環境差をユーザ間で吸収し、ネットワーク全体で公平かつ快適なゲーム展開を保障する必要がある。また C/S 型通信時にサーバ側で

管理されていたユーザのゲーム履歴データは各ユーザ端末ごとに管理する必要がある。

P2P 型通信はブートストラップ段階の接続手法によってさらに 2 種類に分類できる。

- ピュア型

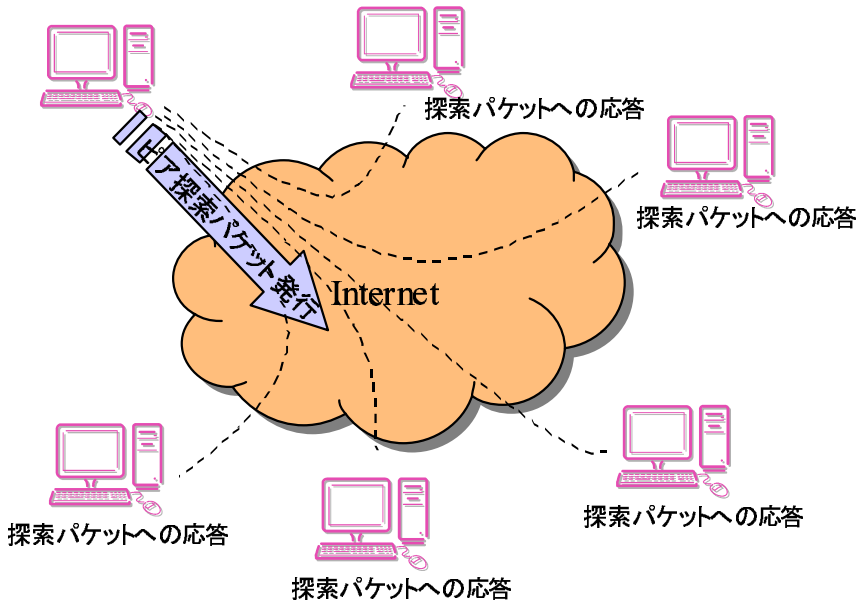

図 2.4: ピュア型 P2P ピア発見概要図

ピュア型 P2P におけるブートストラップ時のトポロジ構築図を図 2.4 に示す。本形式はランデブー型とも呼ばれるが、トポロジに参加するノードは他のノードを探索するパケットをブロードキャストし、発見を試みる。もしトポロジに参加しているノードを発見できた場合は、そのノードが持つ他のノードへのポインタを受け取りさらに接続をする。これを繰り返すことで接続ノード数を増加させていく。ブートストラップ段階から中央管理をする端末が必要ないため完全分散型であり、トポロジ構築段階でサーバを設置する必要がない。ただしサービスに参加しているノードの有無や、発見可能であるかは不透明なため接続までのコストがかかる。

- ハイブリッド型

  ハイブリッド型 P2P におけるブートストラップ時のトポロジ構築図を図 2.5 に示す。ピュア型に対して本形式では P2P のノードリストを管理する中央サーバが存在し、新規に参加するノードはそのサーバへノードリストの要求パケットを

発行する。そしてサーバからリストを受け取り次第、他のノードへの接続を試みる。中央管理するサーバの設置コストが発生するが、サーバが参加者のリストを正確に管理しているため接続が瞬時に実現できる。

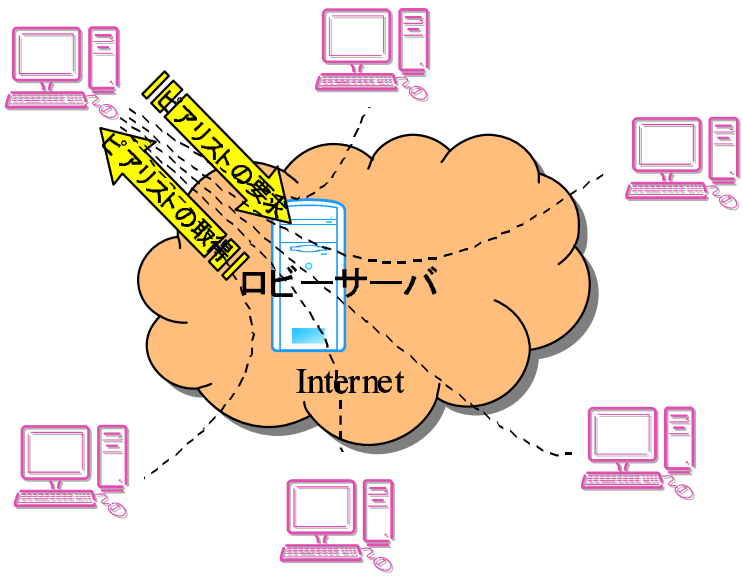

図 2.5: ハイブリッド型 P2P ピア発見概要図

2.1.3 通信の必要要件

実際にゲームを多人数で行う上で、各ユーザの操作をゲーム空間に反映するためにメッセージ通信が行われる。各ユーザの画面への更新はメッセージ到達時に行われるためメッセージが到達する時間がかかるほど、画面への操作結果の反映が遅れる。これがユーザの体感へ不快感を引き起こすことや、メッセージ到着時間差からユーザ間のゲーム状態におけるずれを発生させる。一般的にユーザが快適にゲームを遊んでいると感じるためには、ユーザの操作が画面へ反映されるまでの時間がある一定時間以内でなければならない。インタラクティブなアプリケーションにおいてユーザが画面の更新に大して何らかの違和感を感じるまでの遅延時間は 100〜150 ミリ秒程度である。

Pantel らは Real-Time Multiplayer Game において遅延が及ぼす影響をユーザの熟練度別に観測し、ユーザの体感に及ぼす効果を調査した結果を述べている [26]。この論文では 500 ミリ秒以上の遅延はゲームを遊ぶ上で許容できない結果に至っている。また

IEEEでは標準的なFPSと類似する軍事シミュレーションプログラムであるDistributed Interactive Simulationで許容される遅延は100〜300ミリ秒までと規定している[2]。この結果は概ねネットワークゲームにおける研究と一致する。

ネットワークゲームにおける許容範囲はジャンルごとにおいて違っているが、Pantelらの報告と一致するように、ゲームの特性からリアルタイムな通信を必要とする度合いにより許容遅延は変わってくる。何度も素早い操作と状態の更新を必要とするFPSにおいては150ミリ秒程度、FPSほど頻度の高い更新を必要としないRTSやMMORPGにおいては500ミリ秒程度である。

ネットワークゲームはユーザのボタンを押す等のインタラクティブな操作毎に更新情報がやりとりされるため、高い通信帯域ではなく、通信頻度が必要とされる。ネットワークゲームではネットワーク上に流れるパケット単位は小さく、頻度が多くなるためバースト性のあるトラフィックが発生しやすい。Claypoolらの報告ではCounter Strikeのクライアントが送信する平均のパケットサイズは160バイトであると示している[11]。

2.1.4 ネットワークゲームにおける問題

ここで今日のネットワークゲーム全般で問題とされる技術課題について述べ、特に本研究が問題とする再利用問題について詳細に述べる。

主に問題とされる項目を4つに分け取り上げる。

- チート問題

  ゲームのプログラム自体を改竄したり、ゲームの履歴データを改竄し不正なゲーム展開が行われてしまうことが従来のコンピュータゲームでは問題の一つだった。ゲームがネットワーク化することで、単一ホスト内のプログラム改竄のみならず、メッセージ通信自体を改竄しゲーム展開を優位にすすめる手法が加えて問題視されている。GauthierDickeyらはネットワークゲームで行われるチート行為をgame, application, protocol, networkというレイヤで分類し、各レイヤで起こり得る事象を5種類に分け述べている[13]。

  またMMORPGのようなゲームでは技術のみならずゲーム内における文化的・社会的側面においても、ゲームルールの隙間を利用し不正なゲーム行為で風紀を乱す問題が深刻化している。

- 遅延による同期問題

  インターネットを介してメッセージ通信を行うゲームでは一意な時間で各ユーザにメッセージが到達するという保証はされていない。これはインターネットがベストエフォートなサービスであるから当然である。File Transfer Protocol (FTP)のようなアプリケーションではパケットデータ到達時間に制約は必要としないが、前述したようにネットワークゲームのようなリアルタイムアプリケーション

ではパケットの到達遅延が画面への反応を遅くするため、ユーザの操作性に影響を及ぼす。また遅延の小さいユーザが遅延の大きいユーザより多く操作をゲームに反映できゲーム中に不平等性を引き起こしてしまう [5]。結果として、これら遅延の時間差によりゲーム状態の更新にずれが発生し各ユーザ間のゲームの一貫性が取れなくなる。

- スケーラビリティ問題

  従来のコンピュータゲームでは物理的にゲームへ同時参加できる人数や機器の制約があった。それらがネットワークを介することで緩和され、クライアントの機器が存在していれば大多数のユーザと同時にゲームができるようになったことがネットワークゲームの利点の一つに挙げられる。しかしネットワークを介した多数のユーザのメッセージ処理を実現しなければ、ゲーム空間を公平に維持できない。特に接続形態に C/S 型を採用する MMORPG では、莫大な参加ユーザ数の要求を処理するためサーバの機能を負荷分散し常時稼動させている。現在ネットワークゲームを遊ぶ人口は急速に普及の一途を辿っており、現状のような負荷分散を持ってしても管理しきれない状態が発生する可能性は高い。

- 再利用問題

  現在人気が上がりつづけるネットワークゲームがある一方で、衰退するネットワークゲームが問題となっている。ネットワークゲームの大半が月額利用料金として毎月一定額をユーザから徴収する方式のため、複数のゲームを長期に渡り遊びつづけるユーザは相当な大作ではない限り存在せず、ユーザ一人あたりに一つの定額ゲームが割り当たるため市場的にはユーザ数を分け合ってしまっている。

  各ネットワークゲームの市場割合を図 2.6 に示す。ファイナルファンタジー IX を始めとする代表的なゲームに関しては 10〜20%の割合を占めているが、Second Life [18] や WW2 は 0.5%程度、Puzzle Pirates [34] に至っては 0.3%しか市場を持たない。ネットワークゲーム市場は伸びつづけてはいるが、同時に人気のあるゲームばかりにユーザが集中する傾向が予測できる。

  人気のなくなったゲームはサーバでの管理費用が赤字となり自ずとサービス停止に追い込まれ、少数ながらもそのゲームを継続的に遊んでいたユーザは一方的にサービス終了を強制される。

2.2 再利用問題

本研究にて対象としている再利用問題について詳細に述べる。再利用を考慮するに当たって 3 つの特徴的な側面、サービス提供形態、ネットワーク接続形態、既存ソフトウェアとの親和性における問題をそれぞれ述べる。

今日の商用ネットワークゲームの大半は C/S 型を採用している。C/S 型ではユーザの認証を一元管理できることや、重要なゲーム空間の計算資源がサーバ側にあるため

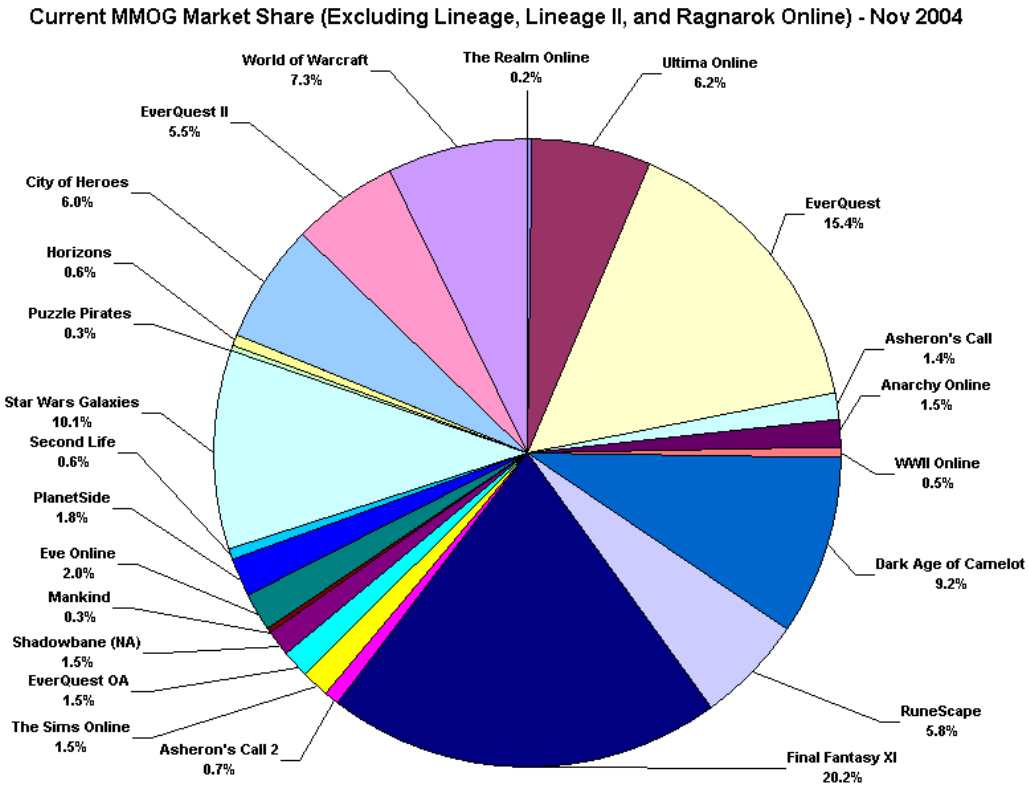

図 2.6: 各ネットワークゲームの市場割合 (出典:MMOGCHART [6])

ユーザ側端末の改造による不正行為を防げるといった利点が挙げられる。またパッチプログラムによる更新があった際に、更新処理を担当するサーバから一括してユーザにプログラムを配布できる。管理コストの面において C/S 型モデルは優れている。しかし現在の C/S 型モデルはサーバを必ず管理しなければならないという点が将来的にネットワークゲームを再利用する場合に問題となる。

2.2.1 サービス提供形態

商用サービスにおいてネットワークゲームの人気が低下した場合に現在ではサービス停止という手段を取るのがほとんどである。しかしゲームはエンタテイメント性を持つソフトウェアであるため、過去のゲームが復刻版として低価格で再販され著しい売上げを収める場合があり、時間が経過しユーザが利用するプラットフォームが進展した後も、再度商品として発売する価値は存在する。しかしネットワークゲームの再利用においては商用ゲームが C/S 型通信形態をとるためサーバ側にゲーム計算資源が集中し、

サービス管理維持費によるリスクを背負うことなくリバイバルの販売戦略を実現できるようにするために、P2P 型通信によるユーザ同士でのサービス提供を簡易に斡旋

できる必要がある。

2.2.2 ネットワーク接続形態

現在の C/S 型モデルの問題としてスケーラビリティの問題が挙げられる。現在の商用ネットワークゲームは大多数のクライアントをサーバ側で処理するため負荷が分散するようにサーバを構成しているが、それでも数十万人規模の同時処理が限界である。規模がさらに大きくなり数百万人規模のネットワークゲームが今度登場しないという保証はない。そのような場合も対処できるよう、P2P 型通信による大規模ネットワークゲームの実現、さらに局所的に C/S 型と P2P 型を組み合わせるようなハイブリッドなネットワーク構成も考慮に入れなければならない。図 2.7 に双方の通信環境を組み合わせて想定されるネットワークトポロジの例を示す。この場合、ゲーム計算を中心とするマスタサーバを用意し、そのマスタサーバへ直接接続するユーザと、局所的に P2P 型通信ネットワークを構築し一部ユーザの端末をゲートウェイとし、マスタサーバへ接続するユーザが存在する。

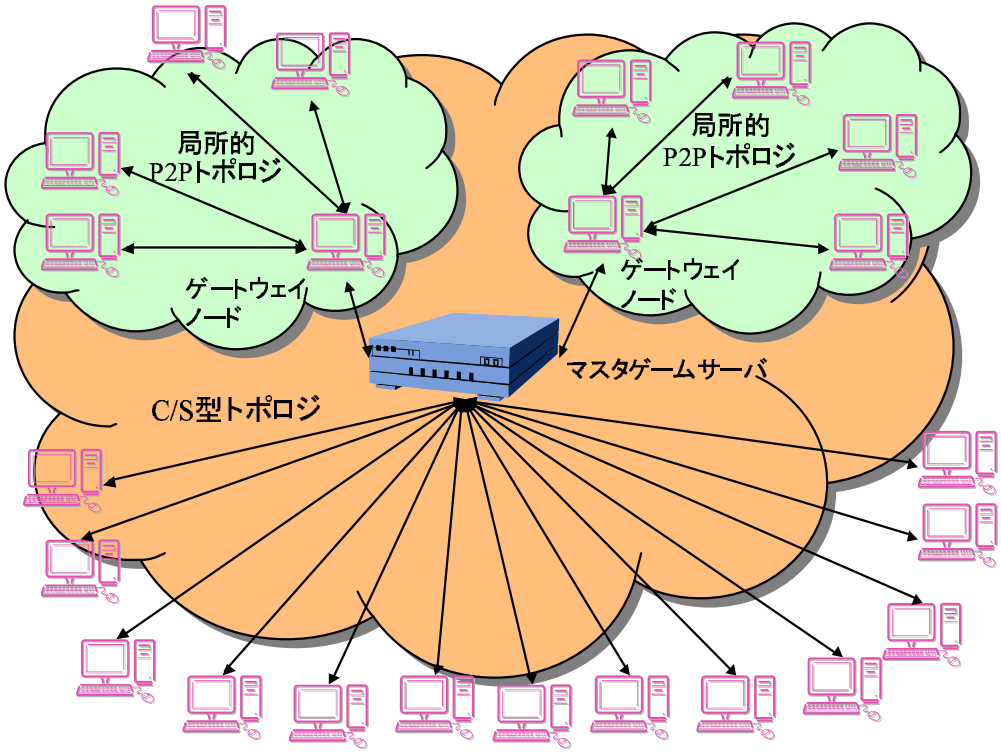

図 2.7: C/S 型・P2P 型相互によるハイブリッド通信トポロジ

P2P 型通信環境に移行する再利用環境においては、遅延問題が C/S 型通信時と比較しさらに深刻になる。なぜならばスケーラビリティを高めるためユーザ間の配送を階

層型にする場合もあるので、メッセージの転送遅延が発生するからである。単一故障点がなくなることは長所であるが、各ユーザの端末側で他のユーザへの経路を正確に管理しなければならない。

2.2.3 既存ソフトウェアとの親和性

通信形態を切り替えネットワークゲームゲームを再利用する場合における一番の問題は、最低でもソフトウェアの通信プログラムを書き換えなければならない点にある。特にゲームアプリケーションにおいては映像描画、ゲーム状態計算処理、サウンド計算処理、通信処理など大規模なプログラム構成になっており、通信部分だけの書き換えも容易なことではない。既存ソフトウェアのコードの改変なく通信形態を切り替えられることが望ましい。以下で既存ソフトウェアを修正しない手法を用いる場合における問題点を述べる。

認証

ソフトウェアの改変なしに通信形態の相互変換を実現する場合にまず問題になるのが、既存ソフトウェアで利用されている認証機能をどのように回避し形態変換するべきかということである。ネットワークゲームは基本的に各ゲームごとに暗号化方法が全く異なる。個々のゲームの認証を解読し、各々のゲーム合わせた認証回避機構の生成は容易ではなく汎用性もない。認証内容自体の解読と改竄を目指すのではなく、認証回避のための汎用的な配送方法を考慮することで既存認証を回避できるような機構が望ましい。

また再利用環境時に実現する P2P 型通信環境においてはサーバ資源が各端末に分散する。これによりサーバプロセスをブラックボックス化することで維持できたゲームメッセージの暗号化が崩れクライアント・サーバ間で通信の改竄が容易に可能となる。つまり P2P 型通信時にメッセージの独自暗号化を提供する必要がある。

メンバ管理

C/S 型通信時には前述したように認証管理サーバ及び計算サーバがゲーム空間に誰が参加しているのか管理する。しかし P2P 型通信時には自律分散型通信になり、中央における管理者がいなくなるため参加ユーザ各自が配送するメンバの情報を管理しなければならない。

トポロジ構築方法

P2P 型ネットワークにおけるトポロジ構築手法に関する研究は今日まで多数行われてきた。主に構築のための要素として、ピアのノードを発見するためのサービス発見機

構と、ピア発見後のネットワークを通信状態に基づいて最適に構築する機構に分かれる。オーバレイネットワークでのサービス発見機構として著名なものとしてChord [33]やTapestry [37]、Pastry [7] [8]といった分散ハッシュテーブル(DHT)を利用したアルゴリズムが挙げられる。これらはトポロジの全経路が判明しない場合において、膨大なノード数が存在するネットワークから少ない探索要求で目的のピアを発見できる。DHTによるアルゴリズムはサービス発見においては効率的であるが、実際のピア同士の通信が最適な経路で行われるわけではない。ピア発見後の最適な経路構築法においてはアプリケーションレベルマルチキャストの研究が挙げられる。本研究分野ではマルチキャストトポロジ構築方法によりメッシュ型トポロジとツリー型トポロジに分類される。一般的にP2P型通信時には参加ユーザ間をメッシュ上に直接接続し配送することは非効率であると既存研究より明らかになっている [7]。ツリー型トポロジの効率的な研究として代表的なものはNarada [14]やScribe [9]が挙げられる。有用性が論文で証明されてきた研究であるため、これらのアルゴリズムを利用し、P2P型通信時におけるルーティングにおいて耐故障性や配送効率を高めることができる。

ゲームゾーン管理

ゲームの再利用時におけるネットワークゲーム独自の問題としてゲームゾーンの管理が挙げられる。ゲームゾーンとはMMORPGで使われる機能で、ゲーム空間が膨大になる場合、一つの計算サーバで全ての空間を含有すると大量のクライアント処理をその一台で行わなければならない。よって大抵のゲームでは計算サーバに担当させるゲーム空間を意味のある領域で分割し、領域が変わる場合はサーバへの接続も変更する手段をとる。

P2P型通信に切り替えた場合、メンバ間に全てのゲームゾーンのデータを流すことは非効率である。そこでユーザの位置するゲームゾーンを把握しておき、同一ゲームゾーンに存在するメンバのみにクライアントアプリケーションが配送を行うといった手法を取る必要がある。

同期管理

C/S型通信においては、ゲーム計算サーバからの更新メッセージが全てのユーザへ1ホップで届く。ユーザの通信帯域や通信状態が異なる場合であっても、サーバで一括して配送回数やタイミングを最適化できる。これによりC/S型では比較的平等なゲーム展開が保障できた。P2P型通信時は各ユーザ間を直接接続するため、遅延やメッセージ損失率といった経路ごとで通信状態に格差が発生する可能性が高い。他の経路と比較し通信帯域が広く、環境が優れている経路が常に有利になるようなゲームルールに起因する要素ではない不公平さは回避しなければならない。そのため通信環境および状態をユーザ間で同一にし、通信環境差による優劣を吸収し扱うことで平等なゲームメッセージ配送を実現する必要がある。

2.3 関連研究

本研究と関連して 4 つの研究及び既存のツールについて述べる。

サーバエミュレーションツール

サーバエミュレーションツールはオープンソースコミュニティ等にて開発されたツールでゲームサーバをエミュレーションしサービス提供側のサーバを介さずにユーザ側で任意のゲームサーバを展開するツールである。

代表的なツールとして bnetd [1] が挙げられる。bnetd は主に Blizzard Entertainment [3] のネットワークゲームを対象に、ゲームサーバの機能をエミュレーションする。本ツールが登場した背景に、過去に Blizzard 社の提供する公式サーバへユーザが多数接続した場合に、サーバの設備が十分でないため負荷が集中しゲームがまともに遊べない状況が多発していた。そのため有志の人々がクライアントおよびサーバアプリケーションの通信内容を解析し、メッセージのプロトコルに合致するエミュレーションプログラムを開発した。これによりユーザが任意のネットワークでサービスを展開し遊べるようになった。尚本ツールを使用する場合、クライアント側は既存のプログラムを変更の必要が全くない。

本関連研究はユーザが任意のネットワークでサービス提供が可能という点において有益なツールである。しかし現在存在するエミュレーションツールは全てサーバ側のプログラムを実装しエミュレーションしているため、ゲームメーカ及びネットワークゲームごとにツールが乱立している。またゲームの種類が増える度に実装しなければならない。そして本ツールは根本的にネットワーク接続形態を変更するものではないため、前述した再利用問題で列挙した課題の解決には至らない。

SimMud

Knutsson らは P2P 型広域通信基盤である Pastry とその上に位置する Scribe システム上に SimMud という Massively Multiplayer Game (MMG) を実装した [16]。本関連研究は設計段階からどのように膨大な仮想空間を MMG に最適な形で分散処理させるかという議論を交えた興味深い研究である。具体的には仮想空間上での位置が近いもの同士が通信を行うという Interest Management の手法を用いてユーザのアバタをゲームマップごとにグループ化し、マップをアバタが移動するとその別のグループに接続を切り替えるという方法をとる。また P2P 環境における同期問題の解決として、状態の相違が起こった場合に coordinator と呼ばれる仲介者に問い合わせることで同期を行う。

この研究においては分散型ネットワークゲームの開発における留意するべき点をまとめている。しかしルーティング方法に Pastry を選んだ理由や、他の分散ハッシュ関数を用いるルーティングアルゴリズムへの適応が容易という記述があるが具体的に変

更するためのコストの議論がなされてはいない。Pastry 以外の分散ルーティング方式が効率的であるネットワークゲームの登場や、今後新しいルーティングアルゴリズムが登場する場合も考えられるため、ゲーム部分とは独立した形でルーティング部が提供されゲームにあったルーティングを選択できることが望ましい。

ネットワークゲーム開発用ライブラリ

Microsoft DirectPlay [19] や Multiterm Massplayer System [22] といった昨今のネットワークゲーム開発向けネットワークライブラリではネットワーク接続形態を透過的に扱える開発ライブラリが登場している。これらを利用することで通信接続形態の変更によるプログラムの修正量は大幅に削減が可能である。しかし前述したようにゲームソフトウェアは膨大な量のプログラムで構成されている。書き換えを必要とする部分が通信部分だけであっても、大半のネットワークゲームを遊ぶだけが目的であるユーザに修正作業を委ねることは現実的ではない。

2.4 本章のまとめ

本章ではネットワークゲームの概要について説明し、ネットワークゲームでの一般的問題について述べた。またその中で本研究が対象とする再利用問題について詳細に説明した。いくつかの関連研究を挙げ、それらを利用しても本研究が対象とする再利用問題についての解決には至らない趣旨を述べた。再利用のための通信接続形態を自由に切り替えるフレームワークは未だ提案されておらず、その開発にあたってはいくつかの多角的視点から考慮すべき点があることも述べた。

第3章　Gaming Swapperの設計

前章にて廃れていくネットワークゲームの増加が問題になることを述べた。人気のないネットワークゲームを維持するために、管理費用を必要とせず運用するにはP2P型通信環境が必要となる。しかし再利用を実現するフレームワークは未だ提案されておらず、再利用の観点によるネットワークゲームの必要要素も不明確である。本章では前章で述べたネットワークゲームの再利用問題を解決するため、C/S型通信をP2P型通信に切り替える機構の設計について詳細に述べる。まず本研究が想定する環境について述べ、設計に当たって留意した点を再利用問題に関連して3つの汎用的な観点から説明する。次に本研究にて提案する機構の具体的な設計を、全体モジュールの概要、構成するモジュールごとの詳細という流れで述べる。

3.1　Gaming Swapper の概要

本機構はネットワークゲームの基盤通信をC/S型とP2P型で相互変換する機構である。変換するに当たり、既存ソフトウェアを書き換える必要がないことを主な特徴とする。システム適応後の概要図を 3.1 に示す。具体的にはC/S型通信時のクライアント・サーバアプリケーションの構成をP2P型通信時においては各クライアントにサーバ機能を含有させる。クライアント及びサーバアプリケーションから生成されるゲームメッセージは全て本機構を通過する。本機構にはゲームアプリケーションとは独立したP2P型通信基盤を提供し、P2P型通信時の各種メッセージ配送支援モジュールを機能に持つ。クライアントからのメッセージは内包するサーバとP2P型通信を行っているメンバが内包するサーバに配送することで、C/S型通信環境を透過的に扱い、下位のP2P型通信環境の状態一致を図る。

3.2　設計方針

本機構の設計における方針と留意点について述べる。

3.2.1　サービス提供形態への柔軟性

従来のC/S型モデルでは既存ソフトウェアを利用したネットワークゲームの再利用時に問題が発生することを述べた。本研究ではサービス提供側による容易な接続形態の切り替えだけでなく、ユーザにとっても容易な接続形態の切り替えるを実現できる

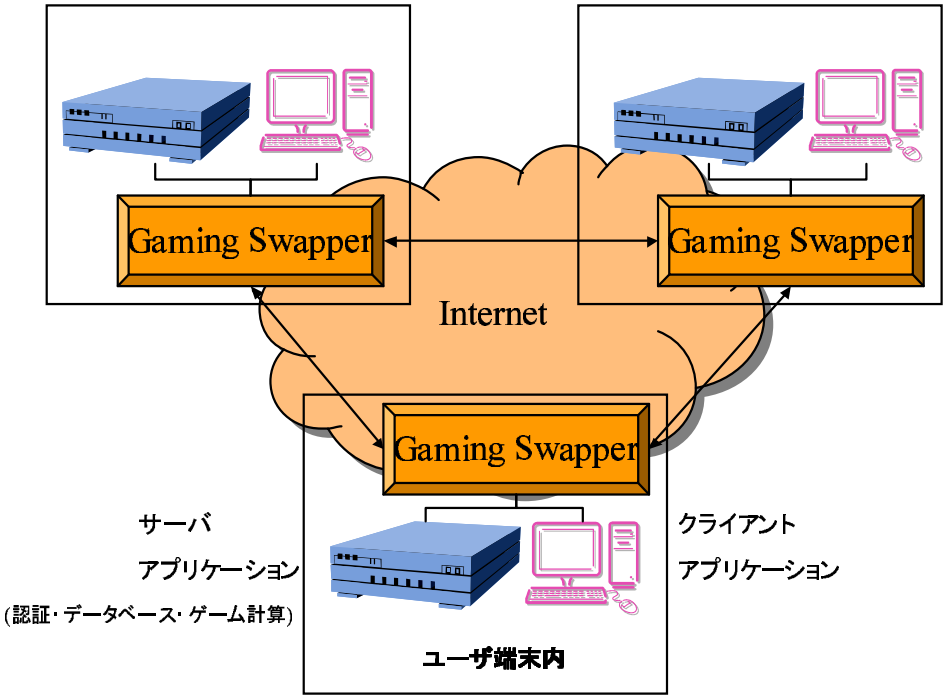

図 3.1: 概要図

機構を目指す。

具体的な設計への反映として、ゲームプログラムが C/S 通信時に利用していたアドレスとポート番号、及び P2P 通信時に接続すべきアドレスとポート番号をルールとして記述することで各種ゲームへの対応を試みる。

3.2.2 ネットワーク接続形態への柔軟性

本機構では再利用問題と関連し、C/S 型通信時におけるスケーラビリティの低さや、前述した C/S 型と P2P 型通信が混合したネットワーク構成に対応するため、根本的に C/S 型通信を維持し、本機構と組み合わせ P2P 型通信を提供する。また P2P 型通信時に変更した場合で発生しうる通信環境差の問題へ対応する機能を本機構に内包し設計する。

3.2.3 既存ソフトウェアへの親和性

通信接続形態を変更する場合に既存ソフトウェアのネットワーク部分の変更が問題の一つであると述べた。本機構はネットワークゲームで提供されているバイナリ自体

を書き換えることなく、パケットの配送において端末内部で C/S 型によるエミュレーション通信を行い、同時にインターネット上において P2P 型通信によるルーティングを実現する。また各種ネットワークゲームに対応するため、ルーティングの設定をルール記述方式にし、記述内容をゲームごとに変更できるものとする。

3.3 想定環境

本機構が対象として想定する環境について述べる。まず本機構を利用する上での前提条件として以下の要素を挙げる。

- サーバ側プログラムの全バイナリが公開されている

  本機構はサーバ側プログラムをクライアントホスト内で同時実行させ、raw ソケットにて取得したパケットのアドレスを変更しサーバプログラムへクライアントのパケットを配送するため、バイナリが公開されている必要がある。

- クライアントが利用する各サーバのアドレスが判明している

  本機構は既存の C/S 型通信を維持し接続形態を切り替えるため C/S 型通信時にクライアントから出されたパケットのうち、本来 C/S 型通信時のサーバに送るべきパケットを内部のプロセス宛てに修正し送ることでパケットのデータ部分を変更することなくクライアントに返すべき正当な結果を得る。そのため配送時のアドレスの修正に反映するために、クライアントプログラムが接続するサーバのアドレスが分かっていることが前提となる。

- サーバ側バイナリが複数存在する場合、アドレスとバイナリの対応が判明している

  前述したようにサーバ側機能が負荷分散などを目的としていくつかのプロセスに分けられている場合、クライアント内部での正当な結果を得るプロセスへのパケット配送を行うため、アドレスとバイナリの対応が判明している必要がある。

- クライアントおよびサーバにおける各プログラムの利用ポートが重複しない

  本機構ではユーザの同一端末内でクライアント及びサーバプログラムを同時稼動させる。この場合それぞれのバイナリへのパケット配送を判断する基準として IP アドレスとポート番号を利用するが、IP アドレスは物理的なインタフェイスにマッピングされる場合が大半である。このような場合ポート番号で配送を判断しなければならないため利用ポートが重複しないことが前提である。

- NPC が登場しないゲームを対象とする

  MMORPG ではゲーム参加ユーザ以外にも、ノンプレイヤキャラクタ (NPC) というゲーム空間に自動で配置されたアバタが存在する。アバタは大抵 C/S 型通信におけるゲーム計算サーバ側で一括計算され、結果のみ各クライアントに配送

される。サーバでは各々で生成される乱数によってNPCの移動座標は計算される。そのためNPCの移動を同一化するために、サーバごとの乱数も同期する必要がある。本機構ではサーバプロセス自体の同期に対する解決は主眼ではなく、今後の課題の一つであるため、前提条件としてNPCの登場は考慮しない。

また本機構はハイブリッドP2P型通信を基本的に対象とし、各ユーザは他の参加ユーザのアドレスを知っているものとする。これは現在存在する商用P2P型ネットワークゲームのトポロジ構築においても、ロビーサーバを設置し、そこに各ユーザが接続した後で、直接ユーザ同士を接続するサービスが多いからである。今後もブートストラップ開始段階においてはハイブリッド型が商用P2Pネットワークゲームの中心になると予測する。

ただしピュアP2P型への適応も考慮に入れ、ブートストラップ部分を他モジュールに対して独立性の高いものとする。

3.4 設計詳細

前述した設計方針と想定環境に基づき本機構の設計について詳細を述べる。まず全体構成を説明し、その後各モジュールの説明に移る。

3.4.1 全体構成

本機構の全体構成図を図3.2に示す。長方形で囲まれた部分を本機構が提供する。大きく分けて認証管理モジュール、ゲームメッセージ送受信モジュール、ゲームゾーン管理モジュール、同期管理モジュールに分かれる。またそれらと連動するデータテーブルとしてサーバ管理表、メンバ管理表が存在する。

前提として各ユーザ端末ごとで本機構が動作し、クライアントアプリケーションとサーバアプリケーションが同時に実行されており、その2者間の通信は全て本機構を通る。まずクライアントアプリケーションを立ち上げゲームを開始する場合、大抵のゲームはサーバとの認証から始める。

クライアントから出た認証用メッセージは本機構の認証管理モジュールに到達する。認証管理モジュールにて内部で同時実行されているサーバプログラムと連携しC/S型通信時における認証を回避するようにメッセージ配送を行う。

認証を終えた後、クライアントはロビーサーバ又はサービス探索パケット発行によるサービス発見を試みてゲームへ参加をする。参加した後のクライアントのメッセージは全てゲームメッセージ送受信モジュールが管理する。

クライアントから生成されるメッセージは内部のサーバに届けられると同時に、メンバ管理表を参照しP2P型通信グループのメンバにも配送される。

配送されたメッセージは他のホスト上にて再度本機構のゲームメッセージ送受信モジュールに渡され、包括するサーバアプリケーションに転送し、計算結果が各々の内

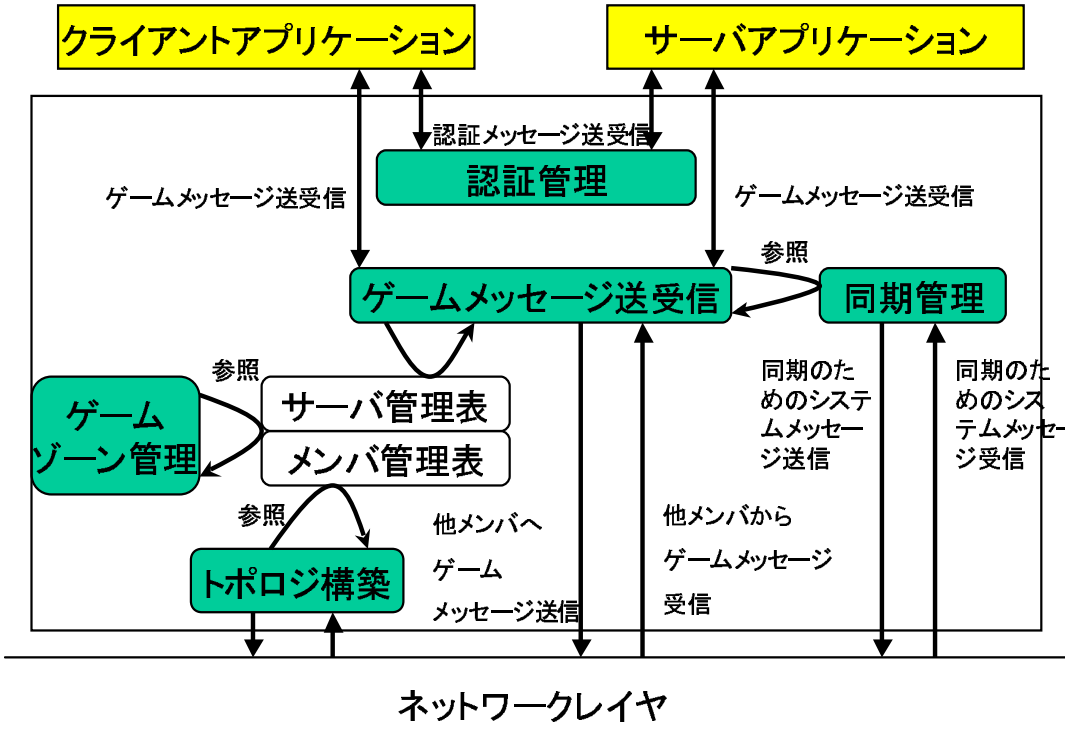

図 3.2: 全体構成図

部クライアントアプリケーションへ反映される。ゲームメッセージの送受信を支援する機構として 3 つの機能が働く。1 つ目はトポロジ構築モジュールであり、これはゲームごとの遅延に対する要求を反映しアプリケーションレベルマルチキャストの構築を動的に最適化する。また 2 つ目の同期管理モジュールでは異種通信環境差を考慮し、ネットワークをモニタリングした結果からメッセージ反映時間を平等にする。そして最後に述べるゲームゾーン管理モジュールでは MMORPG のようなゾーンによって接続するサーバが分割される場合に、同じゾーンにいるメンバにメッセージ配送することで余計なトラフィックがネットワークに流れることを防ぐ。

3.5 モジュールの設計

本節では本機構を構成する 3 つのモジュールの設計と連結して機能する 2 つのデータ表について詳細に述べる。

3.5.1 認証管理モジュール

認証管理モジュールは本機構開始時に機能する。商用ネットワークゲームはほぼ必ずゲーム開始時にネットワークを介しユーザが正規のログイン名とパスワードで接続要求しているか判断を行う。そのためソフトウェアを書き換えず独自のメンバ間ルーティングを実現する上でも各ユーザ端末ごとで既存の認証を通過しなければならない。

また各種ネットワークゲームに対応するために、認証時の配送をルール記述方式にする。ルール記述における要素例を表 3.1 に示す。

表 3.1: 認証管理モジュール設定ファイル記述要素

項目	内容	備考
クライアント側認証ポート	クライアントアプリケーションが認証に利用するポート番号	
サーバ側認証ポート	サーバアプリケーションが認証に利用するポート	
認証サーバ対応バイナリ	認証機能に利用するサーバアプリケーションのバイナリファイル名	
データベースサーバ対応バイナリ	認証時のデータベース管理を行うサーバアプリケーションのバイナリファイル名	
ハイブリッド接続情報	C/S 型及び P2P 型通信を同時に行い認証する場合の対応アドレスと対応ポート番号	将来的な拡張を考慮したオプション項目

バイナリを再利用する場合に考慮すべき配送ルール記述が最小限になるよう設定した。しかし上記記述内容においても設定をユーザが書き換えることは困難な場合がある。本機構の利用にあたってはサービス提供側が予め用意したルール記述ファイルをネットワークを介し配送することで、ユーザが全く設定を変更せずに利用可能なことも有効な方法として想定する。

3.5.2 ゲームメッセージ送受信モジュール

単一端末内で C/S 型通信のゲームメッセージ配送及び P2P 型通信時にメンバ間でゲームメッセージの配送を行うゲームメッセージ送受信モジュールについて述べる。

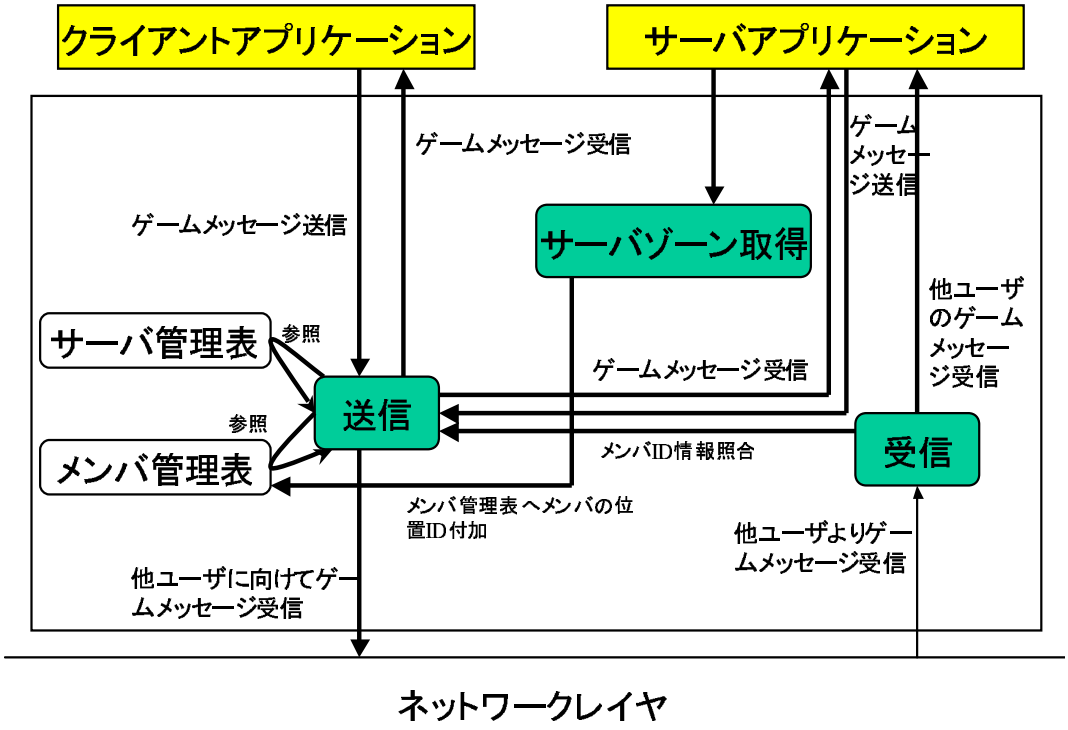

図 3.3: ゲームメッセージ送受信モジュールにおけるデータフロー図

クライアント及びサーバアプリケーションを含めたゲームメッセージ送受信モジュールにおけるデータフローを図 3.3 に示す。ユーザが何らかの操作を行い、その更新情報がクライアントアプリケーションからメッセージとして生成される。

生成されたメッセージは本モジュールの送信部に渡される。送信部では渡されたメッセージの IP アドレスとポート番号を参照し、サーバ管理表と対応を調べる。

もしメッセージが内部で起動しているサーバアプリケーション向けであることが判明した場合、アドレスをそのプロセス向けに修正し、配送する。

計算サーバのプロセスでクライアントからのメッセージが計算され、その反映結果がまたメッセージとしてサーバから生成される。生成されたメッセージは送信部へと渡され、前述と同様に内部のクライアントプロセス向けにアドレスとポート番号を修正した後配送される。

また送信部にてクライアントから生成されたメッセージが計算サーバに向かうと同時に、P2P 参加メンバにも送信される。送信すべきメンバはメンバ管理表に記述してあり、それを参照する。

P2P 参加メンバにて送信されたメッセージは本機構の受信部に到着する。受信部では内包する計算サーバプロセスへメッセージを配送する。この時メッセージに後述するゾーン ID が付与されていた場合、送信部へメッセージを転送し、メンバ管理表の

ゾーン ID テーブルを更新する。

3.5.3 トポロジ構築モジュール

トポロジ構築モジュールでは P2P 型通信時におけるメンバ間のネットワークトポロジを動的に変化させ遅延や帯域に基づく最適化を図る。前述したようにアプリケーションレベルマルチキャストの手法は多数提案されているため、それらアルゴリズムを組み込み連携し機能する。

また本論文ではネットワークゲーム向けアプリケーションレベルマルチキャストの最適化機能として遅延に基づく階層型経路構最適化手法について提案しトポロジ構築モジュールの 1 つとして組み込む。

3.5.4 ゲームゾーン管理モジュール

本モジュールは MMORPG で使用する手法であるゲーム計算サーバ側の膨大なゲームゾーンが、意味のある領域でプロセスごとに分割された場合を想定する。MMORPG では同じゾーンにいるユーザが同一のサーバへ接続する。この時、他のゾーンで局所的に発生したメッセージはゲームゾーン全体に影響するメッセージ以外受信する必要がない。

しかし P2P 型通信環境においてユーザ同士を直接接続する場合、全てのクライアントのメッセージがゾーンに関係なく配送される可能性が考えられる。

そこで P2P 型通信時の無駄なメッセージトラフィックを削減する手法として仮想空間上の位置が近いユーザ同士のみがメッセージ交換を行う Interest Management という手法が提案されている [28]。

本機構ではサーバプロセスごとにゾーンが分かれているという前提を利用し Interest Management を行う。具体的手法としてゲーム計算サーバプロセスのバイナリ名とデータサイズからゾーン ID を生成し、ゲーム開始時にメンバ管理表のアドレスに対応づけることで関係しないゾーンにいるユーザ同士がゲームメッセージを配送するのを防ぐ。

また本モジュールは基本的に MMORPG を対象としているが、大規模な FPS や RTS における適応も可能とする。

3.5.5 同期管理モジュール

ネットワークゲームにおいてはユーザ間で公平にゲームを遊ぶため同期処理が重要であることは述べた。P2P 型通信においては分散通信環境になるため各ユーザ端末ごとに通信環境が異なり、遅延変動が激しくメッセージ到達時間の差が出る。よって同期処理は C/S 型通信時より難しい問題となる。

同期処理の問題は解決が難しいことが既存の研究より判明している。Lin らは C/S 型通信のネットワークゲームにおいて、クライアントとサーバ間のメッセージを実時

間通りに処理させるのではなく、各端末ごとの遅延に合わせて遅らせた後で反映させる Sync-in・Sync-out のメカニズムを用いて同期処理を実現するフレームワークを提案した [17]。しかし本関連研究においても各端末の時計が GPS 等により高精度で同期していることを前提としており、十分な実用性に耐えうるかどうかは疑問がある。

本機構では絶対時間を利用しない同期手法を提案する。まず同期管理モジュールから遅延計測のための微小パケットを定期的に P2P 型通信参加メンバに送信し、計測の履歴情報から基づいたメンバごとの平均到達時間を算出する。そして各メンバへのメッセージ送信時間を、最も遅延のあるメンバに合うように平均到達時間分ずらすことで平等なメッセージ配送を試みる。

3.5.6 サーバ管理表

サーバ管理表では主にゲームメッセージ送信モジュールと連結し C/S 通信時に利用していたゲーム計算サーバ群のアドレス、ポート番号、バイナリ情報を記述する。送信部において一番始めに参照され、アドレス修正時に使われる。

サーバ管理表におけるルール記述項目を表 3.2 に示す。

表 3.2: サーバ管理表設定記述要素

項目	内容	備考
クライアントポート番号	クライアントプロセスが利用しているポート番号	
計算サーバアドレス	ゲーム空間の計算を行うサーバのアドレス	複数ある場合はポート番号、バイナリ名とペアで羅列
計算サーバポート番号	計算サーバが利用するポート番号	複数ある場合はアドレス、ポート番号とペアで羅列
計算サーババイナリ名	計算サーバが表示しているプロセス名	複数ある場合はアドレス、ポート番号とペアで羅列

記述する内容は計算サーバが利用しているアドレス、ポート番号、バイナリ名である。また MMORPG のように計算サーバが複数存在する場合はそれぞれをペアにして羅列する。具体的な記述内容については次章にて示す。

3.5.7 メンバ管理表

メンバ管理表はP2P型通信時の参加メンバを管理する。本機構はハイブリッドP2P型を想定しているため、ロビーサーバにて初期接続を相互ユーザ間で行うため、そのロビーサーバの情報を設定ファイルに記述しておく。送信モジュールがそれを認識し、サーバへの接続を行う。

またユーザが本機構を利用してP2P型通信によるゲームを行う場合にロビーサーバを必要とせず、相手のアドレスが分かっている既知の人と行う場合も考えられる。本機構ではその場合に対応するため、ルール記述にロビーサーバを介さない主旨を記述し、P2P参加メンバの情報を羅列することで直接接続しネットワークを構築可能とする。

メンバ管理表におけるルール記述の必須項目を表3.3に示す。

表 3.3: メンバ管理表設定記述要素

項目	内容	備考
ルーティングアルゴリズム	利用するルーティングアルゴリズムの選択	
認証サーバアドレス	P2P 型通信時にゲーム開始時のロビーとなる認証サーバのアドレス	
初期ノードアドレス	DHT を利用するルーティングアルゴリズムにおける初期接続ノードのアドレス	オプション記述で認証サーバを介さない場合に利用
メンバアドレス	P2P 型通信時の参加メンバのアドレス	オプション記述で DHT を利用しないメンバ間直接接続を行う場合に利用
メンバポート	P2P 型通信時の参加メンバへ配送するポート番号	

3.6 ネットワークゲーム用メッシュ型アプリケーションレベルマルチキャスト最適化機能

本研究ではトポロジ構築モジュールにてアプリケーションレベルマルチキャスト(ALM)の最適化を動的に行うことを述べた。現在までに多数の最適化手法が検討されている。本論文においてもネットワークゲーム向けメッシュ型ALM最適化アルゴリズムを提案しトポロジ構築モジュールへ組み込む。

3.6.1 ALM トポロジの分類

ALM はアプリケーションレベルでの経路として捉えるため、論理的なネットワークトポロジを考慮する。ALM のトポロジには大きく分類しメッシュ型ネットワークとツリー型ネットワークが存在する。メッシュ型のトポロジ例を下記に示す。

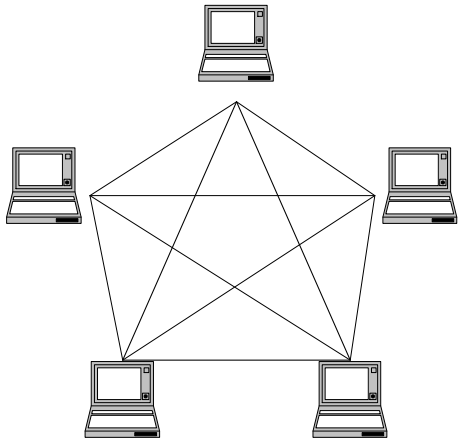

図 3.4: メッシュ型トポロジモデル

現在存在する P2P 型ネットワークゲームにおいてはユーザ同士の発見はハイブリッドで行われる場合が多い。また論理ネットワークトポロジには図 3.4 のようなメッシュ型モデルが利用される。

3.6.2 メッシュ型トポロジにおける遅延の問題

メッシュ型トポロジにおいては、ALM によりユーザ同士が各々ユニキャストで接続する。この場合、遅延の大きい通信経路を利用する可能性がノード数が増えるにつれて高くなる。遅延の大きい経路を利用しつづける場合、一部のノードでゲーム状態の整合性が取れない状況が起こりうる。また他のノードを経由した方が遅延が少ない経路が存在する場合も考えられる。

3.6.3 ANGEL の設計

メッシュ型トポロジにおける遅延に基づくトポロジ最適化機構として ANGEL(Architecture for Network Games utilizing Eligible Leaders) を提案する。本機能はメッシュ型トポロジ構築後、参加ノードが遅延に基づき分類される。遅延情報を管理するノードを立ち上げ、遅延情報を収集する。もし直接接続された経路より他のノードを経由した方が遅延が少ない場合は、管理ノードが経路切り替えの指示を行いトポロジを最適化する。

ANGEL の構成図を図 3.5 に示す。まずハイブリッド型によるノード同士の接続を行いメッシュを構築する。その後遅延計測機能により各ノードが接続しているノード

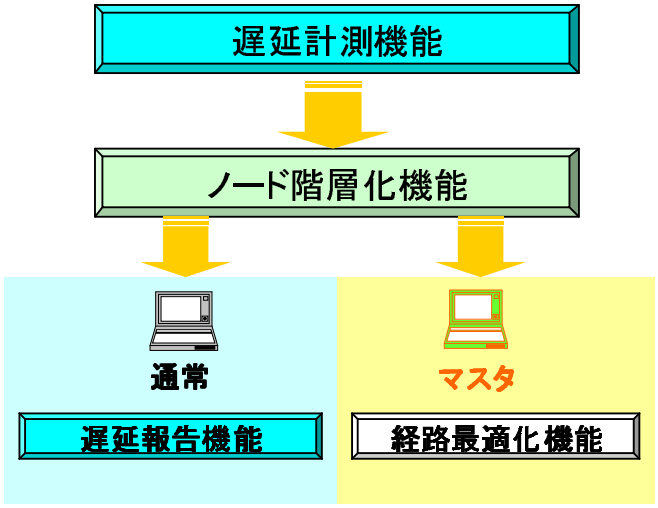

図 3.5: ANGEL 構成図

に対して遅延計測を行い、そこからノード階層化機能により通常ノード (ON)・マスタノード (MN)・ルートノード (RN) に役割が分類される。分類されたノードのうち、ON は遅延報告機能により直接接続している通信経路の遅延を、遅延計測する間隔 (遅延計測間隔) に基づいて計測し MN に報告する。MN は ON からの遅延情報に基づいて経路ツリーを構築し、より低遅延な経路を発見した場合、経路切替メッセージを ON に送信する。これを持続的に行うことで経路の最適化を図る。

以下、各機能について詳細に述べていく。

- 遅延計測機能

  各ノードが直接接続しているノードとの遅延を計測し、そこから平均遅延と最大遅延を計算し、隣接するノードと交換を行う。

- ノード階層化機能

  遅延計測機能によって計測された平均遅延を交換した結果、最も低い値を持つノードが自律的に MN として起動する。平均遅延が同じである場合は最大遅延から判断する。MN は起動した後、起動していることを知らせる間隔 (マスター確認応答間隔) に基づき確認応答を ON に送り、ON が MN が落ちたことを検知した場合、再度選出する。また MN が複数存在する場合、同様にして MN 内から RN を選択する。

- 遅延報告機能

MN 起動後、それ以外のノードは ON として、直接接続している隣接ノードに対して遅延計測間隔で遅延の計測を行い、遅延の値を遅延テーブルに保持する。ON は遅延テーブルの情報を、遅延を報告する間隔 (遅延報告間隔) で MN に送信する。

- 経路最適化機能

  MN が各ノードから集めた遅延情報から経路ツリーを作成する [14]。そしてより遅延の少ない経路を発見する間隔 (経路探索間隔) に基づいて経路ツリーを探索し、より遅延の少ない経路が発見された場合、経路情報から経路切替メッセージを送信する。各ノードはそれに基づいてアプリケーションレベルで通信経路を変更する。

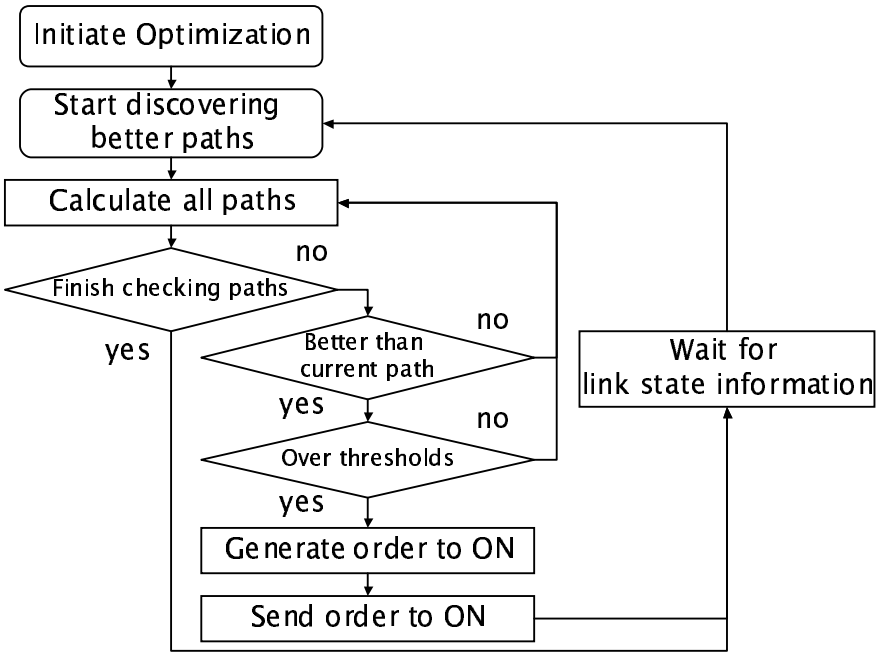

図 3.6: 最適化アルゴリズムフロー図

図 3.6 にマスタノードにおける最適化アルゴリズムの流れを示す。経路探索間隔の回数に達する度にマスタノードにて図 3.6 に示すアルゴリズムを実行する。まず各ノードが遅延報告間隔で送信し収集した経路情報の結果からダイクストラ法を用いてより遅延の少ない経路が存在するかどうか計算する。もしそういった経路を発見した場合は次に経路切り替えの閾値を超えているか比較する。超えていなかった場合は経路探索に戻るが、超えていた場合は切り替え対象のノードへの切り替え命令生成部に移る。そして生成した命令を対象のノードへ送る。命令を受信したノードは内容に基づきより最適な経路へと切り替えを行う。

3.7　本章のまとめ

本章ではネットワークゲームにおける再利用問題とスケーラビリティの問題を解決する機構である Gaming Swapper の提案と設計を行った。まず本機構が動作する想定環境を述べ、ソフトウェアの設計において考慮した点を前章に対する解決方法として述べた。本研究ではサービス提供者、ユーザともに利用しやすい形態であり、かつ多様なネットワークゲームへの対応を意識し、ルール記述による通信形態切り替えを実現する。本機構は 5 つのモジュールから主に構成されており、それぞれの機能について詳細に説明した。またトポロジ構築モジュールでのアルゴリズムにおいて、本論文ではネットワークゲーム向けメッシュ型アプリケーションレベルマルチキャスト最適化アルゴリズム (ANGEL) を提案し、組み込むことを述べた。

第4章　Gaming Swapper の実装

4.1　実装の概要

本章では本機構の実装について詳細に述べる。本論文ではプロトタイプを実機上にて実装した。まず実装を行った環境について説明し、次に実装レイヤの考察に関する議論と選択理由を述べる。そして実装したモジュールについて述べる。

4.1.1　実装環境

開発はプログラミング言語に C++を用いて Windows2000 上に実装した。実装においてはパケット取得ライブラリとして WinPcap ライブラリ [12] を利用した。WinPcap ライブラリは Windows プラットフォーム向けの Pcap ライブラリ環境である。UNIX 系 OS で利用されるネットワークパケットフィルタである Berkley Packet Filter (BPF) を利用した Pcap ライブラリと高い互換性を持つ。そのため、今後本機構をマルチプラットフォームで実現する場合にプログラムを根本から作り直すような大幅な実装修正を必要とせず、比較的容易に各種 UNIX 系 OS に移植することが可能である。

またプラットフォームに Windows を選択した理由として、OS の利用者が多く、多数のネットワークゲームが発売されているため需要が他のプラットフォームに比べ多いと予測するからである。

4.1.2　レイヤの選択

実装するレイヤの選択として主に以下の選択が挙げられる。ここではまずそれぞれの利点と欠点について述べ、本機構を実装する上で選択したレイヤと選択理由について述べる。

- アプリケーションレイヤでの実装

  ユーザレベルで動作するアプリケーションとして実装する方法がある。カーネルには手を加えず実装可能であるため、利用する手続きにおける汎用性や各種 OS で実現する場合の移植性に優れている。しかし一般的にカーネルレベルで実装された場合より処理にかかるオーバヘッドが大きい。

- ネットワークレイヤでの実装

ユーザレベル以外での方法としてソケット層やセッション層で実装する方法が考えられる。この場合アプリケーションで実装する場合より処理にかかるオーバヘッドが少ない。しかしカーネルモジュールなど、OS に手を加えることになるため汎用性が低く、移植性がない。

ネットワークゲームは今日、Windows、Linux、そして PlayStation2 [31] や GameCube [25] 等の家庭用コンソールとマルチプラットフォームで登場している。本機構をネットワークレイヤにて実現する場合、プラットフォームの OS ごとに実装していくことはコストも大きく、またユーザ自身が導入を行う場合ネットワークゲームを遊ぶ全てのユーザに対し、OS の変更を強制することは現実的ではない。そして今後も新しいプラットフォームが登場することは容易に予測されるため汎用的な形態であることが望ましい。以上のことを考慮し、アプリケーションとして実装する。

4.2 モジュールの実装

本論文では提案する機構のプロトタイプを実装した。また各種ルール記述ファイルの記述方式には、汎用的な記述方式である XML [36] を採用した。

4.2.1 モジュール実装の概要

ここでは実装した各モジュールの説明を行う。実装したモジュールは認証管理モジュールとゲームメッセージ送受信モジュール、そしてそれらと連結して機能するサーバ管理表とメンバ管理表である。

4.2.2 認証管理モジュール

認証管理モジュールでは C/S 型通信時の認証を通過させるため、ルールに基づいてメッセージの内部転送を行う。動作原理については次に述べるゲームメッセージ送受信モジュールと類似の機構を使用する。

また認証管理モジュールにおける設定の具体的な記述内容例を表 4.1 次に示す。

表 4.1: 認証管理モジュール表記述 XML

```
<certificationforwarding>
  <client>
    <port>10000</port>
  </client>
```

```
<certification>
    <server>
        <ip>133.27.XX.XX</ip>
        <port>10010</port>
        <process></process>
    </server>
<certification>

<database>
    <server>
        <ip>133.27.XX.XX</ip>
        <port>10020</port>
        <process></process>
    </server>
</database>
</certificationforwarding>
```

certificationforwarding タグに囲まれた部分が設定として読み込まれる。client タグはクライアントアプリケーションが利用する情報を記述する。port タグ部分に認証で利用するポート番号が入る。

続いて各認証情報が記述される。各タグの意味は certification タグが認証サーバ情報、database がデータベースサーバとなっている。各サーバ情報には必ず ip タグ、port タグ、process タグがあり、IP アドレス、ポート番号、バイナリプロセス名に対応している。

本プロトタイプでは各項目を必ず記述することを前提とする。

4.2.3 ゲームメッセージ送受信モジュール

ゲームメッセージ送受信モジュールでは、ユーザ端末内部でのプロセス間のメッセージ転送と P2P 型通信間のメッセージ送信を扱う。まずユーザ端末内部での処理について説明し、次に P2P 型通信配送処理を説明する。

表 4.2: 内部メッセージ転送擬似コード

```
void InternalCaptureAndSender(){
```

```
        CaptureBuffer buffer;
        MatchAddress addr;

        buffer = CapturePacket();
        if ((addr = MatchAddress(buffer)) == TRUE){
            TranslatePacketAddress(addr, buffer);
            SendPacket(buffer);
        }

    }
```

表 4.2 に内部での転送処理における擬似コードを示す。まず CapturePacket に示すよう、WinPcap ライブラリを利用してパケットをバッファに取得する。MatchAddress でルール記述内容と取得したバッファを比較し、適合するアドレスおよびポート番号があるか判断する。もし適合する情報があった場合、TranslatePacketAddress でそのアドレスとポート番号にバッファの情報を書き換える。そして SendPacket にて対象とするプロセスに向けてメッセージが送信される。

表 4.3: ピアメッセージ送信擬似コード

```
void PeerToPeerSender(CaptureBuffer buffer){

        MemberTable table;
        DestinationAddress dstAddr;

        while(table != NULL){
            dstAddr = NextDestination(table);
            TranslatePacketAddress(dstAddr, buffer);
            SendPacket(buffer);
        }
}
```

P2P 型通信時のピアへのメッセージ送信処理における擬似コードを表 4.3 に示す。WinPcap にて取得したパケットを引き継ぎ処理に入る。P2P 型通信時の参加メンバ情報は MemberTable に格納されている。各メンバへの送信ごとにバッファのアドレスを更新し、メッセージを送信する。この時パケットのデータ部分への改竄は行わない。

4.2.4 サーバ管理表

サーバ管理表の記述例を表 4.4 に示す。

表 4.4: サーバ管理表記述 XML

```
<internalforwarding>
  <client>
    <port>20000</port>
  </client>

  <calculation>
    <server>
      <ip>133.27.XX.XX</ip>
      <port>20010</port>
      <process></process>
    </server>
  </calculation>

  <calculation>
    <server>
      <ip>133.27.XX.XX</ip>
      <port>20020</port>
      <process></process>
    </server>
  </calculation>
</internalforwarding>
```

サーバ管理表の設定内容は internalforwarding タグで囲まれる。client タグは各ユーザが起動しているクライアントアプリケーションに関する情報を記述する。ここではクライアントアプリケーションが利用するポート番号を port タグに記述する。

4.2.5 メンバ管理表

メンバ管理表の記述例を表 4.5 に示す。

表 4.5: メンバ管理表記述 XML

```
<gamerouting>
  <peermatching>
    <ip>133.27.XX.XX</ip>
    <port>30000</port>
```

```
    </peermatching>

    <member>
        <ip>133.27.XX.XX</ip>
        <port>30010</port>
    </member>

    <member>
        <ip>133.27.XX.XX</ip>
        <port>30011</port>
    </member>
</gamerouting>
```

サーバ管理表の内容は gamerouting タグに囲まれた部分に記述していく。peermatching タグには P2P 型通信時の認証サーバの情報を記述する。サービス参加者を検索する場合は、ここに記述したサーバを利用し行う。ip タグと port タグには P2P 通信時における認証サーバのアドレスとポート番号を入力する。

member タグは P2P 型通信時の 1 参加者を示しており、それぞれの member タグには IP アドレスを示す ip タグと配送すべきポート番号を記した port タグがある。

ポート番号はクライアントおよびサーバアプリケーションと重複しない番号を選択する必要がある。

4.3 本章のまとめ

本章では前章にて述べた設計に基づき、本機構のプロトタイプの実装について詳細を述べた。本機構はマルチプラットフォームでの実現を想定し、ユーザレベルのアプリケーションとして動作する。また実装には汎用的なパケットフィルタリングツールである WinPcap ライブラリを使用した。プロトタイプではメッセージ送受信モジュールと配送ルール記述部を実装した。またルール記述方式として、汎用的な記述方式として利用されている XML を採用した。

第5章　Gaming Swapper の評価

5.1　評価概要

本機構の評価として定量的評価と定性的評価を行う。定量的評価ではプロトタイプ実装のオーバヘッドとメッシュ型 ALM 最適化アルゴリズムの検証を行う。また定性的評価では 2 章にて述べた関連研究と比較を行う。

5.2　定量的評価

本機構の評価を大きく 2 つの観点から行う。1 つ目は本機構を使用した場合と使用しなかった場合のメッセージ処理遅延であり、2 つ目は本論文にて提案するネットワークゲーム向けアプリケーションレベルマルチキャスト最適化機能の評価である。まず本機構におけるメッセージ処理についての評価内容と評価結果を述べ、次にアプリケーションレベルマルチキャスト最適化機能について同様に述べる。

5.2.1　本機構適応による処理遅延

本機構はユーザ端末内で C/S 型通信環境を構築し、同時にインターネット間通信において P2P 型通信を実現する。そのため端末内部で配送サーバプロセスおよび配送メンバのアドレスを参照し変換した後にメッセージを送受信している。定量的評価では本機構が及ぼす配送情報の参照とアドレス変換処理の処理遅延を計測する。

評価環境

本機構を単一のユーザ端末で起動し、同時にゲームプログラムを模した連続的な UDP トラフィックを発生させるクライアントアプリケーションとその結果を返すサーバアプリケーションを起動する。この 2 つのアプリケーションはインターネット上での通信を想定し、接続先のアドレスが固定で組み込まれている。そのため本機構を利用し、クライアントアプリケーションから生成されたメッセージを端末内のサーバアプリケーションに転送するため、サーバ管理表を参照し受信 IP アドレスを端末のアドレス、配送すべきポート番号をサーバアプリケーションが起動している番号に変換する。

尚評価実験に用いた環境は Windows XP を搭載したデスクトップ PC である。

評価軸

本機構を用いた場合にメッセージ遅延がどの程度発生するか計測する。評価はクライアントアプリケーションからの生成メッセージを対象とし、アドレス変換前からネットワークレイヤに出力された直後までの時間を記録する。

評価

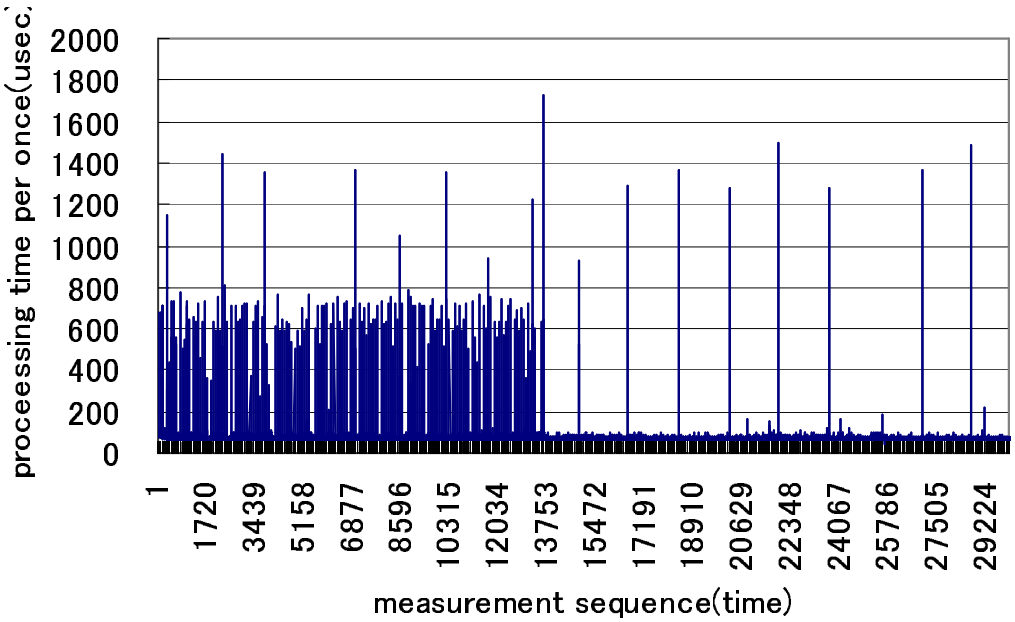

図 5.1: メッセージ処理遅延

メッセージ処理遅延の評価結果を 5.1 に示す。横軸はメッセージ取得開始からの計測回数である。取得開始から約 13000 回目までは 700 マイクロ秒かかっているが、その後は安定して 100 マイクロ秒程度に収まっている。

ネットワークゲームにおけるユーザが感じる不快な体感遅延は前述したように 100 ミリ秒以上において起こる。実際にインターネット上で動作させ、ネットワーク遅延を利用して計測しなければ正確に言及できないが、本機構のメッセージ処理遅延は体感遅延に対して十分微量な範囲である。

5.2.2 トポロジ構築最適化機構の評価

次に本論文にて提案したトポロジ構築最適化機能について評価を行う。

評価環境

評価環境を図 5.2 に示す。3 台のノート PC を dummynet [27] に接続し、各経路でネットワークゲームで発生しうる 200 ミリ秒、40 ミリ秒、40 ミリ秒弱の遅延を発生させた。

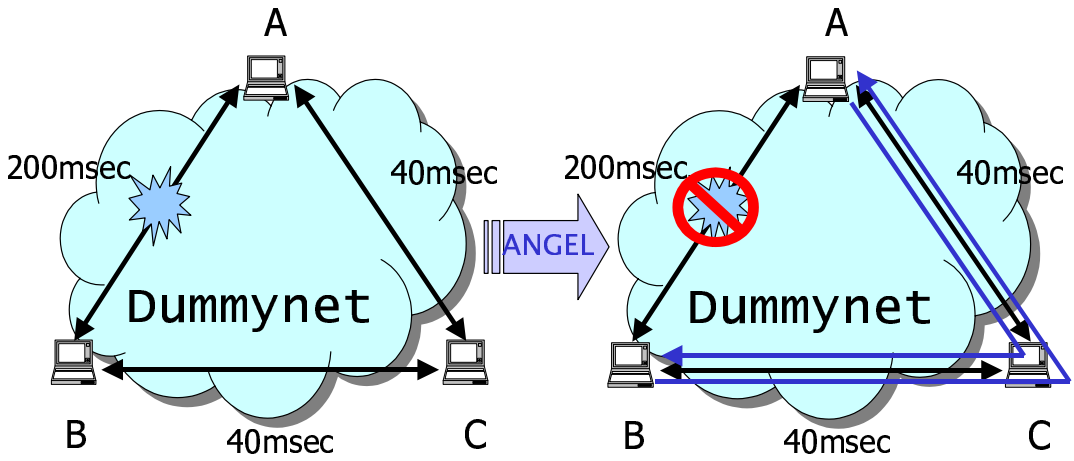

図 5.2: 評価環境と ANGEL 適応図

本機能全体の一部をプロトタイプとして実装した。プロトタイプは遅延計測機能・遅延報告機能・経路最適化機能を有する。そして 2 種類の評価用アプリケーションを作成し評価を行った。アプリケーションは平面上でキャラクタが動き、移動毎にその座標が各ユーザに UDP で送られるプログラムである。2 種類のアプリケーションの違いを下記に示す。

- メッシュ接続型 P2P アバタ移動アプリケーション (メッシュ型)

  トポロジ参加時に全てのユーザとアプリケーションレベルで直接メッセージを送受信する。ネットワークトポロジがフルメッシュになる。

- 最適化機能内蔵 P2P アバタ移動アプリケーション (最適化内蔵型)

  トポロジ参加時に全てのユーザと直接メッセージを送受信し始めるが送受信 API に最適化機能を使用しており、原理に基づき遅延とメッセージ損失率を監視し、経路切り替えを行う。

尚、上記 2 種類の評価用アプリケーションは送受信 API 以外を全て同一のコードを利用した。また評価時の動作を公平にするため、アバタの操作をスクリプトで記述することで自動化し、同一の移動パターンに統一した。そして各ノードの遅延計測メッセージを送信する間隔を 100 ミリ秒から 1000 ミリ秒まで変化させ各計測間隔ごとに 15 回ずつ計測し平均値を算出した。1 回の計測で行った時間は実際に短時間で遊ぶゲームを想定し、10 分とした。

評価で利用した各設定値について述べる。各ノードの遅延報告間隔を 5 回、マスタノードにおける経路探索間隔の回数も 5 回毎に設定した。またマスタノードでの経路最適化アルゴリズムにおける切り替え判断の閾値として遅延が 200 ミリ秒またはメッセージ損失率が 50%を超えた場合に経路を切り替えるよう定めた。

評価軸

評価は経路収束時間と制御通信増加量について行う。経路収束時間とは本最適化機能が遅延計測に基づき、より最短の経路を発見し経路切り替えメッセージがマスタノードで発行されたときの時間とする。また制御通信増加量は評価用アプリケーションにおいてメッシュ型に対する最適化内蔵型の受信データ量増加率を量った。受信データ量は 3 台で計測し合計の差を求めている。

評価

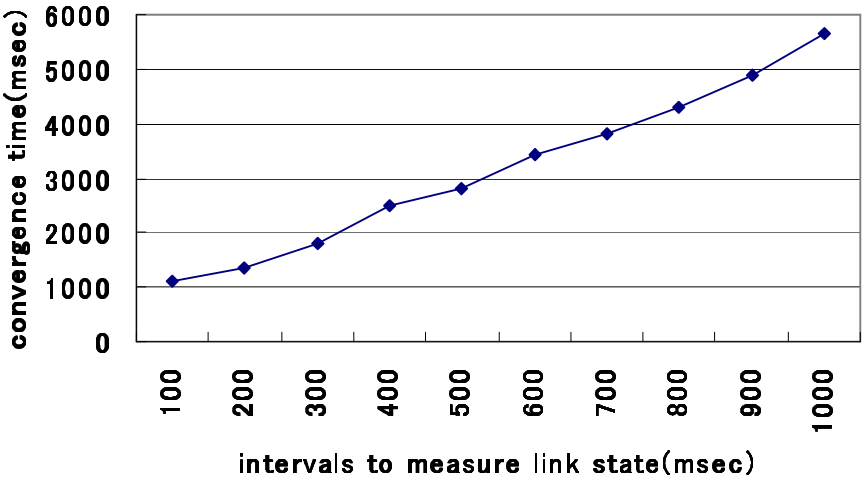

図 5.3: 経路収束時間

経路収束時間の結果を図 5.3 に示す。前述した評価環境から予測できるように、およそ遅延計測間隔×遅延報告間隔から算出される時間で経路の切り替えメッセージが発行された。

また今回計測した環境では計測間隔 1 秒時において経路切り替えメッセージが発行される時間が最高約 5 秒半かかる。主要 FPS クライアントのゲーム参加時間を 1 週間に渡って計測した Chang らの研究では、平均接続時間が 15 から 30 分であると述べている [10]。このことから 5 秒半は割合として微小であり、ゲーム中の大半の時間は最適化された状態で進められると判断する。

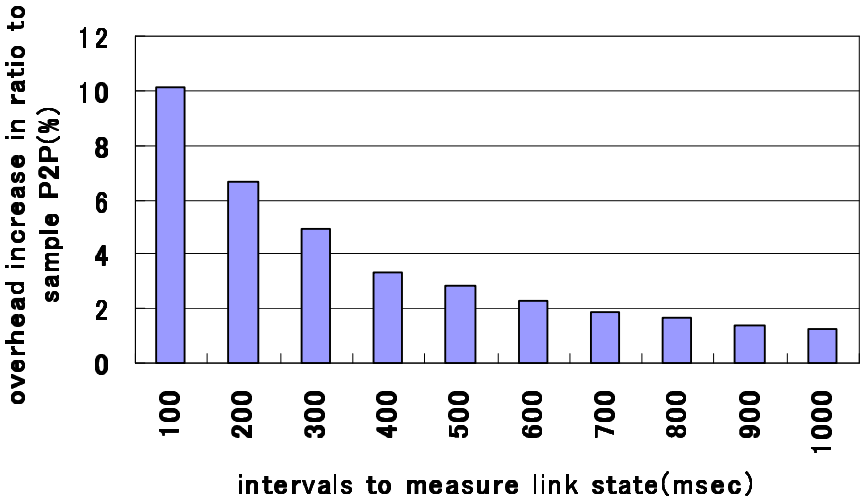

図 5.4: 制御通信増加率

次に制御通信増加率のグラフを図 5.4 に示す。通信量は計測間隔が 1 秒の時に最大で 11%増加する。しかし今回のサンプルアプリケーションではユーザ 1 人あたり毎秒約 1500 バイトのデータを送信しており、人気 FPS である Counter-Strike では毎秒約 3200 バイト送信することが分かっている [11]。そのため一般的ネットワークゲームに本機能を採用した場合、本機能の制御通信増加率の割合はより抑えられると予測する。

5.3 定性的評価

次に本機構を用いた手法と他の手法とを比較した定性的評価について述べる。評価する項目はネットワークゲーム特有の項目と本研究が対象として取り上げる要素を抽出した。評価項目について詳しく述べる。

- コード修正の必要性

  通信接続形態を切り替える場合に既存のソフトウェアの修正を必要とするかどうかに比較する。

- 耐故障性

  ネットワークゲームを運営する上でネットワークトポロジにおいて単一故障点となる部分が存在するかどうか比較する。

- 配送方法の柔軟性

  ゲームメッセージを配送するトポロジ構築方法およびトポロジ維持方法に選択する柔軟性があるかどうか本機構と関連研究を比較する。

- 対応ゲーム種

  通信接続形態の切り替えを行う場合に、本機構を利用した場合と関連研究を利用した場合で全ネットワークゲーム種に対応できるかどうかを比較する。

表 5.1: 関連研究との比較評価

項目	サーバエミュレーション ツール	ネットワークゲーム開発用ライブラリ	SimMud	本機構
コードの修正が不必要	×	×	×	○
耐故障性	×	○	○	○
配送方法の柔軟性	×	○	×	○
対応ゲーム種	○	○	×	×

比較評価した結果を表 5.1 に示す。コード修正の必要性に関しては本機構では前提条件に基づくことで通信接続形態の切り替えを修正なしで実現できるため、他の手法に比べ有効である。また通信接続形態を切り替えることでネットワークトポロジにおいて P2P 型通信になり単一故障点がなくなると言える。またマルチキャスト構築部を独立させているためメッセージ配送方法を複数から選択できる柔軟性がある。しかし本機構では NPC の移動が計算サーバの乱数で行われる場合に対応できないため、全てのゲームへ適応することはできない。

5.4 本章のまとめ

本章では本論文にて提案する機構とネットワークゲーム向けアプリケーションレベルマルチキャストの最適化機能について試験的な評価を行い結果を述べた。評価から単一ホストにおいてゲームメッセージがネットワークへ流れるまでの遅延がわかり、ネットワークゲーム全般において許容できる範囲であることがわかった。またアプリケーションレベルマルチキャストの最適化機能については本実験環境においては有用性を示すことができたが、本提案手法ではネットワーク上に流れるトラフィック量にある程度の負荷がかかることがわかった。

第6章　結論

最後に本研究のまとめを述べる。まず今後の課題を述べ、次に本論文のまとめを述べる。

6.1　今後の課題

本研究の今後の課題として主な点を列挙し、その詳細について述べる。

実機への実装

本論文においては提案した機構を実機で実装を行い、ゲームメッセージ処理時のオーバヘッドを評価した。しかし利用したネットワーク規模は現実的なものではなく、今後大規模なネットワークトポロジにおいて動作させた場合の問題点を考慮するため、シミュレータ上への実装、またはインターネット上での評価を行い、本研究の実用性及び有効な環境を正確に規定しなければならない。

ルール記述における適合性の検証

本研究では多種ネットワークゲームに本機構を適応するため、ゲームごとにルーティングのためのルール記述を行う。しかし記述内容が全てのネットワークゲームに完全対応してる保証はない。今後それぞれのネットワークゲームが利用する通信情報を調べ、より柔軟なルール記述内容を確立する必要がある。また記述方式について本機構利用者によりわかりやすい方式を再考しなければならない。

同期問題

本機構には設計として同期モジュールを含んでいるが、具体的なモジュールの機能については言及しなかった。同期問題はネットワークゲームにおいて難解な問題の 1 つであり、今後関連研究を調べ、自律分散環境に合った同期機能を同期管理モジュールに組み込み、その効果を評価する必要がある。

サーバごとの乱数による計算処理への対応

想定環境に NPC などのサーバごとに乱数で計算される要素は前提としないことを述べた。これは乱数で発生する値がサーバ間で異なるため、同期させることが難しい点に起因する。しかし MMORPG や C/S 型 FPS、C/S 型 RTS にはサーバで乱数計算させる要素が含まれる場合がある。本機構の自律分散環境においても対応できるようにしなければ全てのゲームに対応することは難しい。今後の課題として考慮し解決を試みていく。

メッシュ型 ALM 最適化アルゴリズムの向上

本論文ではメッシュ型 ALM の問題を述べ、また解決する機構として ANGEL を提案した。ANGEL は本想定シナリオを用いた実験環境において有効性を示すことができた。しかしより汎用的なシナリオを用いた場合や最悪のシナリオを想定した場合における有効な評価を行っていない。

6.2 まとめ

本研究では、ネットワークゲームの再利用を可能にするための機構を提案した。人気がなくなり廃れていくネットワークゲームは、現在深刻な問題であり莫大な開発費を投資して発売するネットワークゲームにビジネスチャンスを与えるためにも再利用のためのフレームワークが必要となる主旨を述べた。そして問題定義に基づき、本機構の設計を行った。設計における留意点として多種ネットワークゲームへの対応できるよう、パケット配送判断手法としてルール記述方式を採用した。また今後の課題として大規模ネットワークトポロジでの評価と改良、実機を利用したインターネット上での評価を進めていく予定である。

謝辞

本研究を進めるにあたり、御指導を頂きました、慶應義塾大学環境情報学部教授徳田英幸博士に深く感謝致します。

また、貴重な御助言を頂きました間 博人氏、斉藤 匡人氏、峰松 美佳氏、高橋 ひとみ氏、滝澤 允氏をはじめとする慶應義塾大学徳田研究室 ECN 研究グループの諸氏、また徳田・村井・楠本・中村・南研究室の方々には、研究会の活動を通じて多くの御意見、御助言を頂きましたこと拝謝します。

そして研究の日々を共に過ごした、青柳 禎矩氏、山下 勝司氏、その他多くの友人に感謝します。

最後に、私が望んだ道に進むことをあらゆる面で支え、今日まで不自由なく生きる環境を与えてくれた家族に心から深謝と敬愛を表し、謝辞と致します。

2005 年 1 月 22 日
金田 裕剛

参照論文

- 金田 裕剛, 峰松 美佳, 斉藤 匡人, 間 博人, 徳田 英幸.
  “P2P ネットワークゲームのための階層型遅延最適化ミドルウェアの提案と実装”
  第 66 回情報処理全国大会, Vol. 66(3) pp. 521-522,
  2004 年 3 月.

- Yugo Kaneda, Mika Minematsu, Masato Saito, Hiroto Aida, Hideyuki Tokuda.
  “ANGEL : A Hierarchical State Synchronization Middleware for Mobile Ad-hoc Group Gaming”
  Second International Conference PERVASIVE 2004 Workshop on Gaming Applications in Pervasive Computing Environments.,
  April. 2004.

- Yugo Kaneda, Mika Minematsu, Masato Saito, Hiroto Aida, Hideyuki Tokuda.
  “ANGEL : A Hierarchical Path Optimization Middleware for Real-Time Multiplayer Gaming in Wired and Wireless Networks”
  第 12 回マルチメディア通信と分散処理ワークショップ, Vol. 2004, pp. 329-334,
  2004 年 12 月.
  Winner of THE YOUNG RESEARCHER'S AWARD

参考文献

[1] *bnetd: a free Battle.net server: http://www.chiark.greenend.org.uk/ owend/free/bnetd.html.*

[2] IEEE Std 1278-1993. IEEE Standard for Distributed Interactive Simulation–Application Protocols (Revision and redesignation of IEEE Std 1278-1993), 2002.

[3] Blizzard Entertainment. *http://www.blizzard.com/.*

[4] Blizzard Entertainment. *Warcraft: http://www.blizzard.com/wow/.*

[5] Jonathan Blow. A LOOK AT LATENCY IN NETWORKED GAMES. *Game Developer Magagine*, 1998.

[6] Bruce Sterling Woodcock. *MMOGCHART: http://www.mmogchart.com/.*

[7] M. Castro. An evaluation of scalable application-level multicast built using peer-to-peer overlay networks, 2003.

[8] M. Castro, P. Druschel, A. Kermarrec, and A. Rowstron. Scalable application-level anycast for highly dynamic groups, 2003.

[9] Miguel Castro, Peter Druschel, Anne-Marie Kermarrec, and Antony Rowstron. Scribe: A large-scale and decentralized application-level multicast infrastructure. *IEEE Journal on Selected Areas in Communication (JSAC)*, Vol. 20, No. 8, October 2002.

[10] Francis Chang and Wu chang Feng. Modeling Player Session Times of On-line Games. In *Proceedings of ACM SIG NetGames*, 2003.

[11] Mark Claypool, David LaPoint, and Josh Winslow. Network Analysis of Counter-strike and Starcraft. In *Proceedings of the 22nd IEEE International Performance, Computing and Communications Conference(IPCCC)*, 2003.

[12] Loris Degioanni, Mario Baldi, Fulvio Risso, and Gianluca Varenni. Profiling and optimization of software-based network-analysis applications. In *SBAC-PAD*, pp. 226–234, 2003.

[13] Chris GauthierDickey, Daniel Zappala, Virginia Lo, and James Marr. Low latency and cheat-proof event ordering for peer-to-peer games. In *Proceedings of the 14th*

international workshop on Network and operating systems support for digital audio and video, pp. 134–139. ACM Press, 2004.

[14] Yang hua Chu, Sanjay G.Rao, Srinivasan Seshan, and Hui Zhang. Enabling Conferencing Applications on the Internet Using an Overlay Multicast Architecture. In *Proceedings of ACM SIGCOMM*, 2001.

[15] id software. *Quake: http://www.idsoftware.com/games/quake/.*

[16] Bj?rn Knutsson, Honghui Lu, Wei Xu, and Bryan Hopkins. Peer-to-peer support for massively multiplayer games. In *Proceedings of INFOCOM 2004*, 2004.

[17] Yow-Jian Lin, Katherine Guo, and Sanjoy Paul. Sync-ms: Synchronized messaging service for real-time multi-player distributed games. In *ICNP '02: Proceedings of the 10th IEEE International Conference on Network Protocols*, pp. 155–164. IEEE Computer Society, 2002.

[18] Linden Research, Inc. *SECOND LIFE: http://secondlife.com/*, 2004.

[19] Microsoft Corporation. *DirectPlay: http://www.microsoft.com/windows/directx/default.aspx.*

[20] Microsoft Corporation. *AGE OF THE EMPIRE: http://www.microsoft.com/games/empires/*, 1997.

[21] Microsoft Corporation. *HALO: http://www.xbox.com/en-US/halo*, 2001.

[22] Multiterm Co.,Ltd. *Massplayer System: http://www.multiterm.co.jp/mps_enterprise.html.*

[23] Nintendo. Famicom Mini: http://www.nintendo.co.jp/n08/fmk/.

[24] Nintendo. http://www.nintendo.co.jp/.

[25] Nintendo. *Game Cube: http://www.nintendo.co.jp/ngc/*, 2001.

[26] Lothar Pantel and Lars C. Wolf. On the impact of delay on real-time multiplayer games. In *Proceedings of the 12th international workshop on Network and operating systems support for digital audio and video*, pp. 23–29. ACM Press, 2002.

[27] Luigi Rizzo. Dummynet: A Simple Approach to the Evaluation of Network Protocols. In *Proceedings of ACM Computer Communication Review*, 1997.

[28] Jouni Smed, Timo Kaukoranta, and Harri Hakonen. A Review on Networking and Multiplayer Computer Games. In *Technical Report 454, Turku Centre for Computer Science*, 2002.

[29] Sony Computer Entertainmen Inc. PlayStation The Best: http://www.jp.playstation.com/Item/Campaign/playstation_the_best.html.

[30] Sony Computer Entertainment Inc. http://www.scei.co.jp/.

[31] Sony Computer Entertainment Inc. *Play Station 2: http://www.playstation.jp/products/hardware/ps2/bbpack.html*, 2001.

[32] SQUARE-ENIX. *FINAL FANTASY XI: http://www.playonline.com/ff11/index.shtml*, 2001.

[33] Ion Stoica, Robert Morris, David Karger, Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable Peer-To-Peer lookup service for internet applications. In *Proceedings of the 2001 ACM SIGCOMM Conference*, pp. 149–160, 2001.

[34] Three Rings Design, Inc. *PUZZLE PIRATES: http://www.puzzlepirates.com/*, 2001.

[35] VALVE SOFTWARE. *Half Life:Counter Strike: http://www.counter-strike.net/*, 2002.

[36] The World Wide Web Consortium (W3C). *Extensible Markup Language: http://www.w3.org/XML/*.

[37] Ben Y. Zhao, Ling Huang, Jeremy Stribling, Anthony D. Joseph, and John D. Kubiatowicz. Exploiting routing redundancy via structured peer-to-peer overlays. In *ICNP '03: Proceedings of the 11th IEEE International Conference on Network Protocols*, p. 246. IEEE Computer Society, 2003.