オーバレイネットワークによる 大規模マルチプレイヤオンラインゲームの開発

西川 憲三† 久保田 光一‡

中央大学大学院 理工学研究科 情報工学専攻†‡

要約: 現在の大規模なオンラインゲームはクライアント・サーバ型で構成されるのが一般的で, 処理がサーバに一極集中し運営が困難になっている. 本研究ではオーバレイネットワークを用いてオンラインゲームをユーザに分散管理させ, 大規模なオンラインゲームを開発する手法を提案する.

キーワード: オンラインゲーム, オーバレイネットワーク, 分散管理

1 序論

1.1 背景

ネットワークのブロードバンド化によりオンラインゲームが一般的になった. 現在のオンラインゲームはクライアント・サーバ型で構成されるのが一般的で, 大量のアクセスがサーバに一極集中し莫大な負荷がかかる. 運営は莫大な負荷に耐えうる広帯域なネットワークの整備や, 高い計算能力を持つサーバのクラスタリングなど非常にコストのかかる敷居の高いものとなっている.

1.2 目的

本研究ではオーバレイネットワークを用いることでオンラインゲームを分散ハッシュテーブルによってユーザに分散管理させ, 大規模なオンラインゲームを開発する手法を提案する.

2 関連する研究

本研究のベースにもなっている飯村卓司氏による “オーバレイネットワークを用いた MOG インフラストラクチャの提案”[1] という研究がある. この研究はゲームのデータや処理そのものを分割しユーザに分担させる, これによりプレイヤはいくつかの必要な管理ノードにアクセスしゲームをプレイすることができる.

3 オンラインゲーム

3.1 魅力

オンラインゲームはネットワークを通じて見ず知らずの遠方の人とも一緒にゲームをすることができる. 定期的にプログラムを更新したり修正パッチを当てることができ, ゲームが更新され続ける楽しみがある.

3.2 特徴

オンラインゲームは一般のネットワークアプリケーションとは違い, ゲーム特有のリアルタイム性を実現するために短時間に細かなデータの通信が行われ, 一貫性を保つためにサーバによって一括して公平に処理される. ゲームの世界に処理しきれない程ユーザ数が増大した場合にはサーバを増設し許容量分に複製する対処法がとられることが一般的である.

3.3 本研究の対象となるゲーム

ゲームにはアクション, RPG, レース, シューティングなど様々な種類がある. オセロ, 将棋などに代表されるユーザが交互に操作するゲームはリアルタイム性を必要としないことから実装が容易であるとされるので本研究では扱わない. 本研究ではリアルタイム性を要求される大規模なオンラインゲームを対象としている.

The Development of Massive Multi Player Online Game by Over Ray NetWork

† Kenzo NISHIKAWA, Information and System Engineering Course, Graduate School of Science and Engineering, CHUO University

‡ Koichi KUBOTA, Information and System Engineering Course, Graduate School of Science and Engineering, CHUO University

4 オーバレイネットワーク

4.1 Peer to Peer

オーバレイネットワークはアプリケーション層で繋がれる P2P(Peer-to-Peer) ネットワークである. クライアント・サーバ型と異なり, 高価な中央サーバを必要とせず容易に柔軟で耐故障性に優れたネットワークを構築できる (図 1).

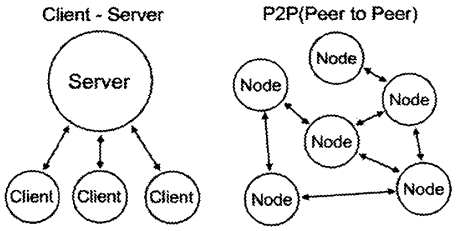

図 1 クライアント・サーバ型と P2P

4.2 分散ハッシュテーブル

分散ハッシュテーブル (Distributed Hash Table; 以下 DHT) は, ハッシュを P2P ネットワークによって分散し管理する技術のことである. 中央のサーバーがなく, すべてのノードが小さい範囲のルーティングと, 小さい 1 セットのデータの記憶を担う. 複数のノードを経由することでネットワーク全体に対してハッシュキーを検索でき, ハッシュ値を管理するノードからハッシュ値の読み込み, 書き込みを可能にする.

4.3 Chord

Chord は DHT アルゴリズムの一種である. ノードを n ビットで表現する ID で割り振り, 昇順で時計周りにリング状に繋ぐ (図 2). キーはハッシュ関数によってなるべく均等に割り当てる. ノードは自身の ID から $2^k(1 \leq k \leq n-1)$ 先の ID に対してショートカットリンクを持つことでノード数 n に対して $O(\log n)$ のホップ数での検索を実現する [2]. 本研究は Chord をベースに改良しリアルタイム性に特化させる.

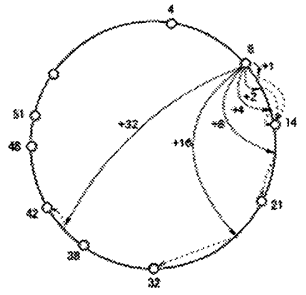

図 2 Chord

5 アルゴリズム

5.1 アプローチ

オンラインゲームで扱うデータは, オフラインで管理するデータとネットワークで共有するデータ (Global Static Data; 以下 GSD) の 2 種類に分けられる. オンラインゲームを大規模化するにはできる限りオフラインでデータを管理し, 少ない GSD を通信することが鍵となる. 本研究では GSD をさらに 2 種類に分けられると考えた. 一時的に共有され破棄されるデー

タと, 永続的にゲームに影響を与えるデータである. 以下前者を AGSD(Auto Global Status Data), 後者を SGSD(Static Global Status Data) と呼ぶ. AGSD は関係するノードの中だけでやりとりされればよいので切り離すことができる. AGSD と SGSD を分担することにより負荷分散を考える.

また, オンラインゲームの実装に必要な項目として次のものが挙げられる.

- ゲームの一貫性を保つため, 分割されたゲーム領域の処理を行うノードは 1 つとする.
- リアルタイム性を実現するため, できる限り短時間で通信が行われる必要がある.
- オンラインゲームの醍醐味である拡張性を持つ.
- 特定のゲームの領域に参加が集中した場合に対応する.

以上の条件を満たすネットワークを考える.

5.2 ゲームのマッピング

ゲームを 1 ノードが処理できる領域に分割する. この作業は動的に行わず予めそのように設計されている必要がある. 分割した領域は Chord に基づきハッシュ関数を用いて均等に近い割合で DHT にマッピングされる (図 3). ゲームを拡張しても同様に効率よく割り当てることができる.

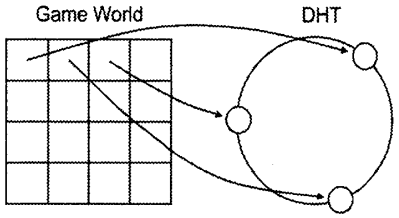

図 3 ゲームのマッピング

5.3 ノードの役割

ノードの役割は次の 3 つに分けられる.

Administrator 領域を管理し, SGSD を処理する.
Owner AGSD を処理する.
Member ゲームをプレイする.

DHT によってゲームの領域を割り当てられたノードはその領域に対して唯一の変更権限を持つ Administrator となる. つまり Administrator は分割した領域の数だけ存在することになる. Administrator は管理する領域に影響を及ぼす SGSD のみを管理し, 子として AGSD を処理する複数の Owner を持つ. さらに Owner は子として複数の Member を持ち, Owner は属するすべての Member に対して AGSD の処理を行う. Owner と Member, Owner と Administrator はネットワーク上で直接繋がれツリー構造となる (図 4). それが Chord と異なる点であり, リアルタイム性を実現する.

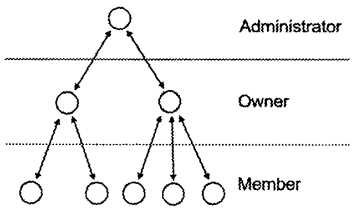

図 4 Administrator と Owner と Member の関係

5.4 ゲームの処理

ゲームを実際にプレイするのは Member である. Member が依頼するデータは前述した通り AGSD と SGSD に分けられる. AGSD は親の Owner によって処理されるが, Owner は複数存在するので Member は Owner ごとにグループ化され, そのグループの Owner によって独立して処理される. これは AGSD が互いのグループに影響しないデータであることに基づいている. また, それとは異なり SGSD はゲームの領域を変更するので全体に影響を与えるデータである. SGSD を処理できるのは唯一の領域の変更権限を持つ Administrator だけであるので, Member の SGSD は親の Owner を通じてさらに親の Administrator に依頼する. そして変更されたデータは Administrator から Owner を通してすべての Member に伝えられる. それぞれのデータは 1 つのノードによって一括して処理されるのでゲームの一貫性を保つことができる (図 5).

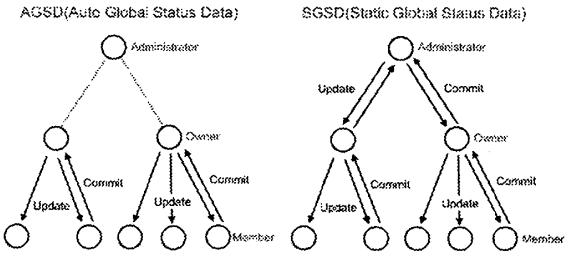

図 5 AGSD と SGSD の処理

5.5 一極集中に対する負荷分散

Administrator が仮に GSD を全て扱うとしても, ゲーム参加者が増え大量の member がノードとして追加されると処理することができない. そこで, 領域に影響を及ぼさない AGSD を切り離し, Owner に命じるようにした. Owner も大量の Member を処理することができないので, ゲームの最大同時接続人数をあらかじめ定めておき, Administrator は特定の人数ごとに Owner を命じて Member を振り分ける (図 6). そうすることで一極集中に対しての負荷を分散することができる.

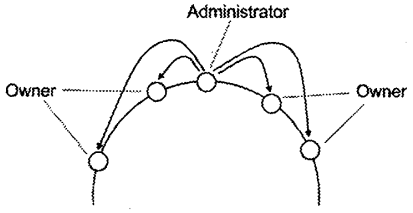

図 6 負荷分散

6 実装

開発は Microsoft Visual C++ の環境下で行い, グラフィック描画 API には DirectX, ネットワーク通信 API には WinSock を用いた.

7 まとめ

本研究では分散ハッシュテーブルの Chord アルゴリズムを改良し, ゲーム特有のリアルタイム性に特化したネットワークを提案した.

8 今後の課題

実装したプログラムのシミュレーションを行い, 性能を評価する予定である.

謝辞

本研究を進めるにあたり研究室の同輩, 後輩にはお世話になった. ここに記して感謝の意を表す.

参考文献

[1] 飯村卓司 “オーバレイネットワークを用いた MOG インフラストラクチャの提案” 奈良先端科学技術大学院大学情報科学研究科修士論文 2005 年 3 月.
[2] I.Stoica, R.Morris, D.Karger, M.F.Kaashoek and H.Balakrishnan: Chord: A Scalable Peertopeer Lookup Service for Internet Applications Proceedings of SIGCOMM'01, pp.149-160, 2001.