グループ通信機構を用いたフォールトトレラントなモービルサービスの検討と実現

長 健二朗
キヤノン株式会社
情報メディア研究所

概要

グループ通信機構とデータレプリケーションを応用したモービルサービスのモデル化と、そのプロトタイプの設計と実装を行なった。

本稿では、移動体通信の制御、特にハンドオフ動作は本質的に多地点間通信であることに着目し、グループ通信機構の提供する信頼性のあるマルチキャストと耐故障性のためのメカニズムが、フォールトトレラントな移動体通信を実現する上で有効であることを示す。

プロトタイプの実装は、グループ通信のプラットフォームとしてIsisシステムをベースにして行なった。

1 はじめに

携帯型計算機と無線ネットワークの普及はコンピュータ通信のユーザ層の裾野を広げるため、分散環境における高可用度(highly available)のサービスの必要性が増してくる。しかしながら、移動体通信において移動中のクリティカルなタイミングでのシステム故障も許容できるようなモデルはあまり知られていない。本研究は、グループ通信機構を用いて、移動端末のサポートと耐故障性の両方を単一のモデルで扱えるシステムの検討と実現を行なったものであり、任意のタイミングによる障害を許容し、かつ、高速でとぎれのないサービスを提供するシステムの構築を目標にしている。

1.1 グループ通信機構とモービルコンピューティング

グループ通信機構[2, 6, 11]は配送順序の保証された信頼性のあるマルチキャストとメンバシップ管理を基本として、分散環境において協調動作するプロセス群をサポートするための通信プリミティブを供給する。そのメカニズムはプロセス間の状態の一貫性の維持に適していることから、主に耐故障性の分野で研究がなされてきたが、近年その一貫性維持の機構が分散環境における広い分野でプログラムモデルの簡素化に利用できることが知られてきた。移動体通信もまた本質的に多地点間通信であるため、グループ通信の分野で研究されてきた成果が応用できる分野である。

本研究では、グループ通信システムのひとつであるIsisシステムをベースに、フォールトトレラントなモービルサービスの構築を検討し、MobileChannelと呼ぶプロトタイプの作成を行なった。

1.2 Isisツールキット

Isisツールキット[3, 4, 5, 6]は、コーネル大学において開発されたシステムで、高可用度のアプリケーション作成のためのツールキットとして位置づけられる。

その特徴は、

- 信頼性(reliability)と配送アトミシティ(delivery atomicity)を持つマルチキャストの実装
- 全順序(total order)、因果関係順序(causal order)、送信順序(FIFO order)の配送を保証するマルチキャストプリミティブの提供
- 停止故障(fail stop)モデルによる故障検出とシステムで一貫した故障レポート
- 動的なグループメンバ管理
- グループ参加時に不可分的(atomic)にアプリケーションの状態移送を行なう手段(state transfer)の提供

を統合することによってバーチャルシンクロニー(virtual synchrony)と呼ばれるプログラムモデルを提供する。このモデルではグループ内のすべてのプロセスが分散環境下のイベントを規定された順序で観測することが保証される。その結果、アプリケーションレベルではステートマシン方式[14]に従ったシステム一貫性維持が可能となる。

ステートマシン方式は、イベントにより決定性(deterministic)状態遷移するプログラムが、等しい初期状態でスタートし同じセットのイベントを同順序で受けとると、均一な状態遷移を行なって均一な状態を維持する性質を利用することによって一貫性を維持する方式である。

Isisの利点は因果関係順序を利用した高速性にある。因果関係順序の保証は、パケットロスがなくメンバの

実行速度差が少ないという通常の実行環境下で順序保証のためのメッセージ交換が不要なため、理論的には配送順序や信頼性を保証しない通信に比べて速度低下がほとんどない。 そのため、Isis は高可用度と高速性を要求される独自のアプリケーション分野を持ち、株式トレーディングや交換機制御等で実用されている。

2 MobileChannel のデザイン

今回設計したプロトタイプ MobileChannel は、移動するクライアントに対して、とぎれのないサービスを提供し、かつ、サーバの故障を隠蔽する機能を持つ。

本システムでは、移動可能なクライアントは一群のサーバによってサポートされる。クライアントは、唯一プライマリサーバと通信を行なうが、すべてのサーバの状態はグループ通信機構によってコンシステントな状態に維持される。プライマリサーバに障害が生じた場合はバックアップサーバの中から代替プライマリが選ばれ、クライアントに対するサービスが継続して行なわれる。

本システムの第一の特徴は、クライアントの移動に伴い意図的にプライマリサーバを切り換えることによって、アクセスポイントがクライアントに追従することである。それにより、本モデルではクライアントのマイグレーションとサーバの障害への対応がプライマリの切り換えという単一の手法で扱える。プロトタイプでは、このシステムモデルを用いて、通信中に動的にプライマリを切り換えるハンドオフ動作が可能な通信チャネルを実装した。

本システムの第二の特徴は、トランスポート層レベルでサーバの複製化を行なう手法をとることによって、簡単かつ高速なハンドオフを実現した通信機構である。

さらに、第三の特徴として、スケーラビリティ向上のために、グループ通信の動的メンバ管理をサーバグループの配置、移動に用いる手法を紹介する。

2.1 システムモデル

本システムは図 1 に示すように移動可能なクライアントと固定のサーバによって構成される。サーバは移動クライアントに対してサービスを提供し、クライアントの代理として動作する。サーバ ·クライアント間の通信には無線通信を想定している。無線ネットワークはセル構造に配置され、それぞれのセルは MSS(Mobile Support Station) と呼ばれるゲートウエイサーバをアクセスポイントとして有線ネットワークに接続される。

セル構造の無線ネットワークで耐故障性を確保する為には、移動クライアントが常に 2 つ以上の MSS にアクセス可能となるように、物理的なレイアウトを工夫する事も必要となる。

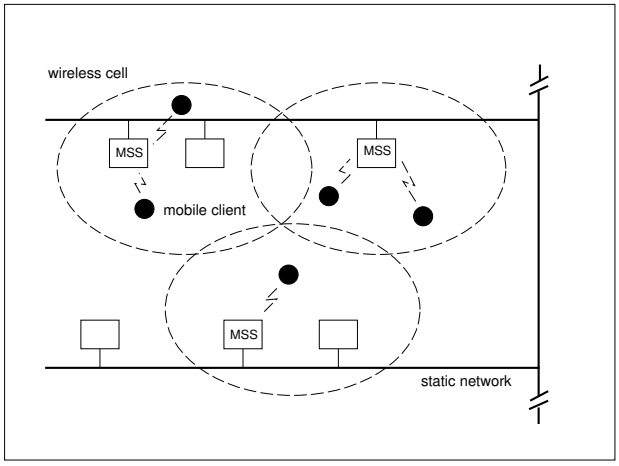

図 1: システムモデル

移動クライアントはサーバに比べ処理能力が小さく、無線ネットワークのバンド幅は有線ネットワークより小さいことを仮定している。本システムではサーバのみがグループ通信を行なうことによって、負荷をサーバ側にシフトし、クライアントの負荷の軽量化をはかる。

主として想定したアプリケーションは、既存の LAN 環境と移動体通信を統合し、かつ、信頼性と応答性の要求されるものである。例えば、工場内を携帯端末を持って移動しながらシステムのチェックやデータ更新を行なうようなものが考えられる。

2.2 プライマリ切り換えによるマイグレーション

分散システムの可用度を上げるためには、サービスの複製化を行ない、故障が発生してもシステム全体が機能を続ける手法が取られる。また、複製化は同時にキャッシュ効果による応答性向上のための手段としても利用できる。

MobileChannel では 2 種類のレプリケーション方式の組み合わせて用いている。ひとつはプライマリ ·バックアップ方式 [8] であり、他方はステートマシン方式である。

プライマリ ·バックアップ方式は、クライアントがプライマリサーバとのみ交信する集中管理方式で、プライマリサーバに障害が生じた場合は、フェイルオーバ (failover) が起こってバックアップサーバのひとつがプライマリに取って代わる。

一方、ステートマシン方式はクライアントがリクエストを全てのサーバにマルチキャストする分散方式である。クライアントからのリクエストに応じて全てのサーバが同期して均一な状態遷移をし、全てのサーバが同一のリプライを返す。クライアントは一番先に届いたリプライを使って処理を続けることができるため、いずれかのサーバに障害が発生しても遅延が発生しない。

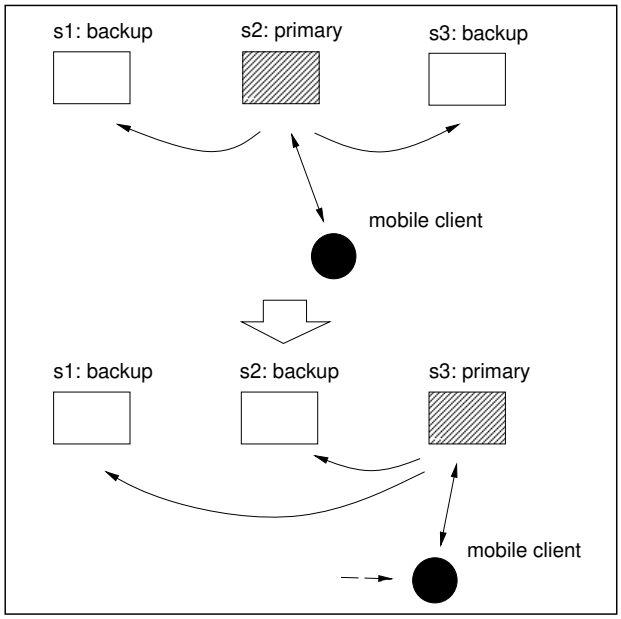

図 2: 意図的なプライマリスイッチ

MobileChannel は、無線の通信範囲の制限と移動クライアントの負荷軽減を考え、ステートマシン方式上にプライマリ・バックアップ方式を組み合わせて使っている。すなわち、移動クライアントはプライマリサーバとしか交信しないが、このプライマリサーバがクライアントからのメッセージを他のバックアップサーバにフォワードすることによって結果的に全てのサーバが等価な状態遷移を行なう。プライマリサーバに障害が発生した場合は、バックアップが代替するが、クライアントにはこのフェイルオーバが透過になる工夫がされている。

MobileChannel では、クライアントのマイグレーションを意図的なプライマリスイッチとして実現している。通常プライマリバックアップ方式ではプライマリの交替はプライマリに障害が発生した場合にしか起こらない。ところが、プライマリを切り換える機能は、移動クライアントに物理的に近い位置にいるバックアップに制御を引き渡すために使うことができる。特にセル構造をもつ無線ネットワークでは、この仕組みによってプライマリを必要な通信範囲に位置させることが可能になる。つまり、図 2 に示すように、位置的に移動クライアントを追従するようにあるサーバから別のサーバへとプライマリの役目を受け渡すことができるのである。

プログラミングモデルとしては、移動クライアントからはひとつの障害フリーなサーバがあるように見える。サーバ側は Isis と同様なステートマシン方式のプログラミングをすることになる。

2.3 トランスポート層のレプリケーション

前節で述べたプライマリの切り換えを利用するには、移動クライアントからシームレスな通信を可能にするトランスポート層の実現が必要となる。MobileChannel では、トランスポート層内に実装されるスライディングウインドウ[1]を複製化して、高速でかつフォールトトレラントな通信を単一の手段で実現している。

従来の移動体通信のハンドオフ手法の多く [1, 12] は、1 対 1 通信の FIFO チャネルを再接続することによってハンドオフを実現している。しかしながら、この手法では、チャネルの再接続時に移動クライアント / 旧サーバー / 新サーバーの三者間での同期とりが必要となり、さらに、同期とりの間に送られたメッセージをバッファリングして配送順序を保証する必要がでてくる。その結果、プロトコルの実装は複雑になり、さらに、ハンドオフ中のクリティカルなタイミングでサーバが故障する場合、なかでも連鎖的に故障が発生する場合への対応は容易ではない。

これらの問題は、グループ通信機構の取り組んできた問題と本質的に同じものである。すなわち、1 対 1 通信のために開発された通信プリミティブを多地点間の通信に用いると、システム全体の一貫性維持が極めて困難になる。グループ通信では、システムレベルでメッセージ配送順序、配送アトミシティ、故障レポートの一貫性を保証することによって、アプリケーションの作成、特に故障時の対応の簡素化をはかる。

MobileChannel では、移動クライアントを軽量化するためにサーバ側のグループ通信機能だけを利用して、1 対 1 の FIFO チャネルは使わず、独自の複製化されたトランスポート層を組み込んだ。1 対 1 の FIFO チャネルを再接続する手法をとると、再接続時にのみ信頼性のあるチャネルの特性が利用できなくなるため困難が生じる。サーバーがコンシステントなスライディングウインドウの複製を持っていれば、クライアントは任意のタイミングで通信相手を切り換える事ができ、かつ、切り換え時に生じる可能性のあるパケットロスは、スライディングウインドウの再送機能で回復できる。つまり、従来手法では切り換えの際にだけ一貫性保持の問題が生じていたが、スライディングウインドウの複製を維持することによって、切り換え時に特別の処理をする必要がなくなり、その結果、ハンドオフ時の同期とりの問題をなくし、かつ、単一の機構で耐故障性を組み込むことが可能となった。

[1] スライディングウインドウ・プロトコルは FIFO チャネルを実現し、通信チャネル内のメッセージフロー制御を行なうために一般に用いられる手法である。

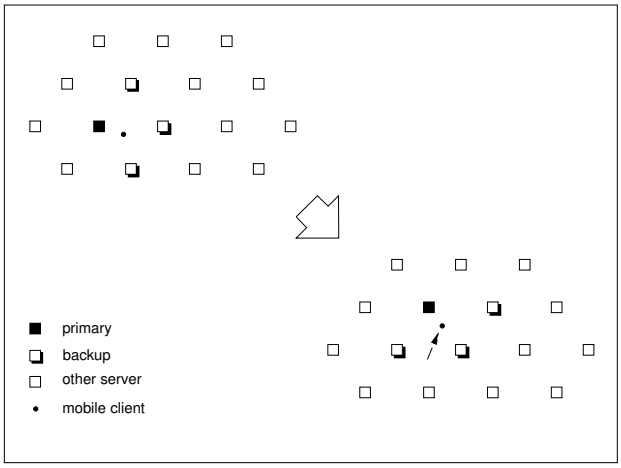

図 3: ユーザサブグループ

2.4 スケーラビリティ

MobileChannel は、グループ通信の動的なメンバ管理をサーバグループの配置、移動に用いることによって、システムのスケーラビリティを上げることを可能とする。

もし、あるユーザの実行状態の複製を、あらかじめ移動する可能性がある全てのサイトに持つ必要があるとしたら、明らかにスケーラビリティ上問題である。通常、ユーザ固有のステートはいくつかのサイトにのみ複製を持つだけで十分である。これを実現するには、サービスを供給するグループが、各ユーザに対応するサブグループを持つ階層グループ構造を持てばよい。ユーザ ·サブグループは位置的にユーザに近いサイトによって構成されることが通信コスト上望ましく、また、セル構造無線ネットワークでは必要となる。

そこで、グループ通信の階層グループによって、このようなサブグループを構成し、動的メンバ管理機能を使ってユーザの移動に伴ってサブグループ自体もユーザに追従するようにすることが可能である。図 3 に示すように、ユーザが移動すると移動方向にあるサイトがこのユーザに対応するサブグループに参加し、後方のメンバサイトがサブグループから抜けるようにする。グループ通信の動的なメンバ管理機構によって、メンバの交替が起こっても全てのメンバが一貫した状態を維持することができる。

また、ディスコネクト状態のクライアントが離れたサイトに再接続した場合には、新規接続ポイントの周辺にサブグループを移動すればよい。

このようなグループの管理は、グループ通信の動的メンバ管理機構を使って、ユーザの位置情報をもとにサーバに指示することによって簡単に行なうことができる。

この方式をさらにスケールアップするには、さらにグループの階層化を進めることで対応できる。現状のグループ通信機構はせいぜい数千サイトまでしかスケールアップしないが、スケーラビリティ向上に関する研究はなされている [10]。

3 実装

MobileChannel はライブラリとして実装され、通信チャネルは UDP 上に実装されている。ライブラリはサーバライブラリとクライアントライブラリに分かれ、移動クライアント管理層と通信チャネル層から構成される。

サーバライブラリは Unix と Isis ツールキット上に作られており、クライアントライブラリは Unix または Microsoft Windows 上で動作する。クライアントライブラリは、Windows 上で約 130KB を占める。このうち約 60KB は、Isis スタイルの構造化メッセージを扱うためのメッセージライブラリである。

現在の開発環境は、Sun SPARCstation とサブノート PC である。サブノート PC は 2Mbps の無線イーサネットカード (NCR WaveLAN PCMCIA) を備え、ブリッジを介して 10Mbps イーサネットに接続されている。しかしながら、我々はユーザの位置検出装置やセルハンドオフをサポートできる無線装置を持っていないため、クライアントのマイグレーションはエミュレータによって起動し、プライマリサーバを切り換えることによって行なった。したがって、物理的に全てのクライアントは同一セル内に留まっており、実際のセル間ハンドオフは実現できていない。

3.1 移動クライアント管理層

クライアント管理層はサーバ側のみに実装され、移動クライアントの管理を行なう。MobileChannel では、マイグレーションはプライマリスイッチによって実現される。ディスコネクト状態のクライアントが別のサーバに接続すると、プライマリスイッチが起こる。ハンドオフは、通信チャネルの切り換えを伴うプライマリスイッチである。また、逆にプライマリのフェイルオーバもマイグレーションとみなすことができる。プライマリ故障時にクライアントが接続状態にあればチャネルスイッチが起こり、これはハンドオフ動作と等価である。

各サーバは mobile view と呼ばれるクライアントの状態リストを持ち、Isis のメッセージオーダリングによって、全てのサーバ上でこのリストは一貫した状態に保たれる。クライアントが最初に接続を行なった時に、新しいエントリが作成され、接続チャネルがこのエントリに登録される。クライアントが接続を終了した

場合、チャネルの登録は抹消されるがエントリ自体は残されて、後にサーバがディスコネクト状態のクライアントの代理として動作するための手段を与える。マイグレーションが起こると、対応するエントリのプライマリを示すフィールドが更新される。

3.2 通信チャネル層

通信チャネル層は切り換え可能な FIFO チャネルを実現する。その実装は 1 対 1 のスライディングウインドウ ·プロトコルの変形で、プライマリのスライディングウインドウの状態をバックアップに複製する手法をとる。

新規にサーバがグループに参加する際には、Isis のステートトランスファ機能によって、アプリケーションの状態に加えて (バッファリングされているメッセージを含む) スライディングウインドウの状態がこの参入サーバに送られ複製が作られる。

プライマリがクライアントからメッセージを受けとると、プライマリは全順序マルチキャストを用いてサーバグループにこのメッセージをフォワードする。したがって、全てのサーバは同じ順序でクライアントからのメッセージを受けとるので複製されたインプットウインドウは等しい状態に保たれる。さらに、サーバはステートマシンとしてプログラムされており、受けとったメッセージに対して全てのサーバは等しい決定性動作を行なう。クライアントへのメッセージの送出はこの均一な動作から行なわれるため、アウトプットウインドウの状態も等しくなる。つまり、サーバがクライアントにメッセージを送る場合は、全てのサーバが同じ内容のメッセージを同じシーケンス番号をつけて自身のアウトプットウインドウにバッファリングする。実際には、プライマリだけがメッセージを送り出すが、これは、サーバ側のチャネルエントリに自分がプライマリであるかどうかを示すフラグがあり、下位層の送出ルーチンはこのフラグをみてフラグがセットされている時のみ送出を行なうようになっているためである。また、下位層の受信ルーチンはこのフラグがセットされているときにのみクライアントからのメッセージを受けとる。プライマリフラグはクライアント管理層によって一貫性が保持され、同時に複数のプライマリが存在しないことが保証される。

アウトプットウインドウにバッファリングされたメッセージは対応したアックを受けとった時点で捨てられる。アックは通常のメッセージにピギーバックされてクライアントから送られ、プライマリによってフォワードされることになる。

アウトプットウインドウの一貫性を保持する上で、全てのサーバーが同一 (identical) な動作をすることは必要条件である。ステートマシンとしてプログラムされていれば、同一順序で同じメッセージが入ってくれば一貫性が保てる。決定性動作を実装する際には、この他に注意すべきイベント要因が 2 つある。ひとつはマシンにローカルなイベントを発生するタイマで、タイマイベントによって複製化された状態を変更すると一貫性が保てなくなる。スライディングウインドウの実装は遅延アックと再送のためにタイマを必要とするが、MobileChannel では、遅延アックや再送動作が直接ウインドウ状態を変更しない事を利用して、これらの動作がウインドウ状態の一貫性に影響を与えないように実装されている。もうひとつの要因は、ローカルなスケジューリングである。システムによっては、ひとつのイベントの処理中にスケジューリングが起こると、残りの処理と次のイベントの処理の順序が保証されない。しかしながら、Isis の場合はノンプリエンプティブ (nonpreemptive) なユーザスレッドが厳密に FIFO 順にスケジューリングされ、かつ、MobileChannel の一貫性保持に関する部分は順序が保証されたイベントによってのみ起動されるよう実装されているため、スケジューリングも一貫性に影響を与えることはない。

また、スライディングウインドウの状態をステートトランスファする時点で、ブロックされているスレッドも問題となる。つまり、ウインドウの状態は転送できるが、ブロックされているスレッドの状態までは転送できないため、このスレッドがチャネルに書込を行なおうとしている場合は一貫性が崩れる。MobileChannel では、ブロックされているスレッドがいる間は、新しいサーバの参入をブロックすることによって、この問題を回避している。

チャネルの接続の管理は TCP モデル [13] にもとづいており、3-way handshake による確立と half-close を使った終了を用いる。TCP の状態もまたステートマシンとして定義されているため整合性がよい。MobileChannel では、クライアントにプライマリスイッチを通知するために、パケットヘッダに SWITCH というフラグを追加拡張した。

図 4 はハンドオフ時のプライマリスイッチとチャネルスイッチ動作を示す。プライマリスイッチは現プライマリにマイグレーション要求をすることによって起動される[2]。プライマリはマイグレーション要求を受けると、まず自身のプライマリフラグをクリアしクライアントとの交信を停止する。次に新プライマリを指定したプライマリスイッチコマンドをサーバグループに

[2] マイグレーション要求はシステムに依存して現プライマリ、新プライマリ、クライアント、あるいは第三者の誰かが行なう。

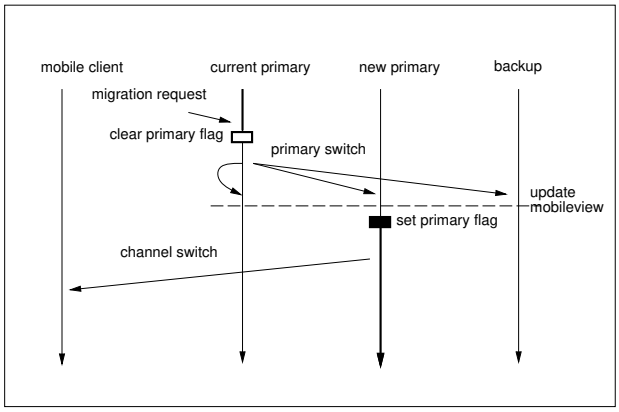

図 4: ハンドオフ動作

マルチキャストする。コマンドを発行したサーバを含むすべてのサーバは、このプライマリスイッチコマンドを受取ると、mobile view の対応するエントリのプライマリ ·フィールドを更新する。ここで、自分が次のプライマリに指定されている場合は、プライマリフラグをセットしてプライマリを代替しプライマリスイッチが完了する。

プライマリスイッチ時に、クライアントが接続状態にある時は、引き続いてチャネルスイッチが実行される。新プライマリは、自分のアドレスとポート番号を指定したSWITCH セグメントをクライアントに送ってプライマリの交替をクライアントに通知する。クライアントはSWITCH セグメントを受けとると、チャネル構造内の交信相手アドレスとポート番号を更新してプライマリスイッチを完了し、この後、新しいプライマリとのみ通信することになる。SWITCH セグメントがロストした時は、通常セグメントと同様に再送が行なわれ、SWITCH セグメントのアックも通常セグメントと同様に行なわれる。なお、セキュリティのため、新規チャネル接続時に生成されるキーを使って、SWITCH セグメント受信時に認証を行なう。

チャネルスイッチの際には、両エンドポイントのウインドウ状態は、シーケンス番号やバッファリングされたメッセージも含め変更されない。したがって、新プライマリとクライアントは、チャネルスイッチの直前の状態から通信を再開することが可能である。異なっているのは、新旧プライマリ上のプライマリフラグとクライアント側の通信相手アドレスだけである。

プライマリが切り換えられる際には、旧プライマリがプライマリフラグをクリアしてから新プライマリがフラグをセットするまでの間、プライマリが存在しない期間があることになる。この間に送られたメッセージはロストすることになるが、チャネルスイッチ完了後通常のスライディングウインドウ機構によって回復される。

通信チャネルの実装において重要なのは、プライマリスイッチは単一のマルチキャストメッセージによって全てのサーバーに報告されることである。Isis の配送アトミシティによって、障害時においても全ての動作中のサーバが全順序でコマンドを受取ることが保証され、すべてのサーバが 1 ステップのアトミック (不可分) な動作としてプライマリスイッチを実行する。さらに、チャネルスイッチも 1 ステップのアトミックな動作である。

また、チャネルスイッチは新プライマリによって実行されるため、フェイルオーバの場合も基本的に同じメカニズムが使える。Isis では、障害レポートは他のメッセージの配送に対して全順序づけされている。したがって、障害レポートを受けとった時点で、バックアップはなんらかの決定性ルールを使って、ローカルな計算で新しいプライマリを選出し mobile view を更新することができる。つまり、障害レポートが図 4 におけるプライマリスイッチコマンドの代りとして機能する。

プライマリとチャネルを切り換える機構はともに、アトミックにシステム状態を更新するが故に、連鎖的な障害にも問題なく動作できる。フォールトトレラントなプログラムを作成する上でこのアトミシティは重要で、動作がアトミックでないと障害の対応は困難なものになる。

現在のプロトタイプの実装では、クライアントは固定アドレスを持っていてチャネルスイッチ後も同じアドレスでアクセスされるが、簡単な変更を加えることでアドレスや通信周波数を変更するように対応できるため、実際の無線通信に適用できる。

MobileChannel では、通信インターフェイスとして、FIFO チャネルの他に、RPC 方式と拡散方式の通信インターフェイスを提供する。これらの方式は、FIFO チャネル上に実装され、プログラミングを簡素化する。RPC 方式では、クライアントはサーバにリクエストを送り、リプライを待ってブロックする。拡散方式は、サーバがすべてのクライアントに対して同じメッセージをマルチキャストする一方向通信である。

この様な通信パターンは多くのアプリケーションに共通のものであり、また、多くのアプリケーションはこの 2 方式を組み合わせて用いる。例えば、株式トレーディングのシステムでは、拡散方式を使って株価情報を通知し、RPC 方式を使って取り引きを行なうことができる。

4 評価

分散環境においてデータの複製を行なう場合、性能低下が大きな懸念となる。しかしながら、レプリケーションを効率よく行なうことは可能であり、我々の実験結果はその可能性を示している。

Isis は複雑なシステムであるが、その通信機能は一般的な通信パターンに対して最適化されているため、現在の Isis のマルチキャストの性能は TCP と大差なく、また、メンバ数１０以下のグループにジョインするのに要する時間は数十ミリ秒である [6, 7]。

先述したように、我々はモービルクライアントの CPU 性能はサーバより低く、かつ、無線通信はサーバ間の通信より低速であると仮定している。この仮定が成り立つとすれば、サーバ側に負荷をシフトしたレプリケーションの仕組みは、実用上クライアントの動作に隠蔽され、性能低下をほとんど招かないことになる。

実験結果によると、MobileChannel のレプリケーションによるオーバーヘッドは小さい[3]。

オーバーヘッドの少ない一因は拡散方式の通信による。拡散方式は RPC 方式に比較してクライアントからのトラフィックを大幅に削減できる。一般的なイベント駆動型アプリケーションにとって、大半のトラフィックは拡散方式でサポート可能なイベント通知である。MobileChannel では、クライアントからのメッセージのみがレプリケーションのためのトラフィック増加につながるため、拡散方式はトラフィックの低減に大きく寄与する。

また、スライディングウインドウの複製化による負荷の増加も、スライディングウインドウの遅延アックが有効に働くためあまり問題とならず、トラフィック増加要因は、拡散方式によるトラフィックの低減より小さいと思われる。

もうひとつの性能低下の懸念として、サーバがグループに参加する際のオーバーヘッドがある。スライディングウインドウの状態移送に関しては、この時点でバッファリングされているメッセージを含むウインドウの全状態がステートトランスファによって転送される。しかしながら、ウインドウのバッファは通常ほとんど空であり、連続通信時でも、２、３個のメッセージがバッファリングされているにすぎないため、転送量は思いのほか小さい。また、転送すべき状態が非常に大きいアプリケーションの場合は、通信のためのプロキシを介する 2 段通信にする構成をとって、プロキシのみを移動するように工夫できる。加えてサーバのグループ参加は、ハンドオフ時のクライアントの移動と比べて、タイミングに余裕がある。

[3] トラフィックは、当然のことながら、アプリケーションの通信パターンに依存する。

user data size [bytes]	4	64	1024
throughput	429	408	300
[msgs/sec]	(220)	(204)	(127)

表 1: Channel Throughput

user data size [bytes]	4	64	1024
1 server	123	122	105
	(56)	(58)	(40)
2 replicated servers	85	95	77
	(54)	(53)	(37)
4 replicated servers	75	73	60
[rpcs/sec]	(45)	(41)	(31)

表 2: RPC Throughput

表 1 と表 2 はプロトタイプの性能データを示したものである。測定は、通常負荷時の 10Mbps Ethernet 上において、サーバ、クライアント共に Sun SPARCstation1 を使ったエミュレーションによって行なった。サーバでは、Isis v3.1 を用いたがハードウエアレベルのマルチキャストは使っていない。かっこ内の値は、クライアントにサブノート PC を用いて、2Mbps の無線 Ethernet を介してアクセスを行なった場合の参考データである。無線通信が約 2 倍遅いのは、ブリッジの介在、二つのネットワークのバンド幅の違い、CPU 間のバイトオーダーの違い等による。特に、無線ネットワークは有線ネットワークの 1/5 しか帯域がないため、高トラフィック時にはブリッジが全てのパケットを転送できない理由で、頻繁なパケットロスが観測された。測定データは、複雑なデータ構造のメッセージ化をサポートする Isis スタイルのメッセージの作成 / 破壊の時間を含んでいる。

表 1 はチャネルスループットを示すもので、サーバからクライアントへの一方向へ連続してメッセージ転送を行なった結果である。表 2 は RPC 方式の通信性能を示すもので、クライアントがリクエストを送って空のリプライを待つ動作を繰り返したものである。RPC 方式の性能は Isis のマルチキャストの性能に支配されている。Isis のマルチキャスト性能は、ハードウエアレベルのマルチキャストを使わない場合ほぼグループサイズに比例する。測定時には IP マルチキャスト環境がなかったが、IP マルチキャストの利用によって大幅な性能向上が見込まれる。

ハンドオフ動作は一般に 20ms 以下で行なうことが可能である。ハンドオフ時間を、旧プライマリがマイグレーション要求を受けてから、クライアントがプライマリを切り換えるまでと考えると、プライマリスイッチコマンドのための全順序の Isis のマルチキャスト (ABCAST) が一回と、チャネルスイッチセグメントのためのパケット転送が一回必要となる。ABCAST は４つのサーバーに対して 10ms 程度、チャネルの１パケット転送は 3ms 程度である。チャネルスイッチセグメント転送中にパケットロスが起こった場合は、スライディングウインドウの再送メカニズムによって再送されるので、通常パケットのパケットロスとほぼ同様の影響を受ける事になる。

今回測定を行なったプロトタイプは機能的に限られたものであるが、その性能は多くのアプリケーションを満足するものであると考えられる。フォールトトレランスと高速なハンドオフを考慮すると、これらの要求を持つアプリケーション分野にとって十分に魅力的なものであると考える。

5 おわりに

今後、携帯型計算機と無線ネットワークの普及により、移動クライアントのためのフォールトトレラントなサービスはますます重要になってくるが、移動ホストや耐故障性への対応は、アプリケーションで個別に対応するには、あまりに複雑な問題を抱えている。一方、これらの問題には、グループ通信の分野で研究されてきた成果がうまく応用でき、システムレベルでより強い一貫性を保証することによって、アプリケーションの作成が容易になる。

本研究では、プライマリの切り換えによってマイグレーションを実現するというモデルを提案し、ハンドオフ動作を簡素化して、かつ、システムの耐故障性を向上させる手法を実現した。本手法は、現在のネットワーク環境やプログラミングモデルにもよく適合し、現在利用可能な技術で実現できる。また、将来的には、グループ通信技術の進歩によって、速度性能やスケーラビリティが向上する恩恵が受けられる。

今後の課題としては、より現実的なアプリケーションを通して、システムへの要求を検討していきたい。

謝辞

本研究は主に筆者が１９９３年から９４年にかけてコーネル大学コンピュータサイエンス学科に研究員として在席している間に行なった研究を基盤としている。その間、Ken Birman 教授を中心とする Isis グループのメンバに数々の貴重な協力と助言をいただいたことを記し感謝する。

参考文献

[1] Arup Acharya and B. R. Badrinath. A framework for delivering multicast messages in networks with mobile hosts. *Technical Report,* Department of Computer Science, Rutgers University, 1994.

[2] Yair Amir, Danny Dolev, Shlomo Kramer and Dalia Malki. Transis: A Communication Sub-System for High Availability. *Technical Report,* CS91-13, The Hebrew University of Jerusalem, Israel, November, 1991.

[3] Kenneth P. Birman. Replication and fault tolerance in the Isis system. *Proceedings of the Tenth Symposium on Operating Systems Principles,* Orcas Island, WA, 1985, 79-86.

[4] Kenneth P. Birman, Thomas Joseph and Frank Schmuck. *Isis—A Distributed Programming Environment, Version 2.1–User's Guide and Reference Manual,* Department of Computer Science, Cornell University, July, 1987.

[5] Kenneth P. Birman, Andre Schiper and Pat Stephenson. Lightweight Causal and Atomic Group Multicast. *ACM Transactions on Computer Systems,* Vol. 9, No 3, August, 1991, 272-314.

[6] Kenneth P. Birman. The Process Group Approach to Reliable Distributed Computing. *Communications of the ACM,* December, 1993.

[7] Kenneth P. Birman and Timothy Clark. Performance of the Isis Distributed Computing Toolkit. *Technical Report,* TR94-1432, Dept. of Computer Science, Cornell University, June 1994.

[8] Navin Budhiraja, Keith Marzullo, Fred B. Schneider and Sam Toueg. The Primary-Backup Approach. *Distributed Systems second edition,* Addison-Wesley, 1993, 199-216.

[9] Kenjiro Cho and Kenneth P. Birman. A Group Communication Approach for Mobile Computing. *Proceedings of Workshop on Mobile Computing Systems and Applications*, IEEE, December 1994, 95-102.

[10] Bradford B. Glade, Kenneth P. Birman, Robert C. B. Cooper and Robbert van Renesse. Light-Weight Process Groups. *Proceedings of the OpenForum '92 Technical Conference,* November 1992, 323-336.

[11] M. Frans Kaashoek and Andrew S. Tanenbaum. Group Communication in the Amoeba Distributed Operating System. *Proceedings of the IEEE International Conference on Distributed Computing,* IEEE, May 1991, 222-230.

[12] Kimberly Keeton, Bruce A. Mah, Srinivasan Seshan, Randy H. Katz and Domenico Ferrari. Providing Connection-Oriented Network Services to Mobile Hosts. *Proceedings of USENIX Symposium on Mobile & Location-Independent Computing,* USENIX, August, 1993, 83-102.

[13] J. B. Postel. Transmission Control Protocol. *RFC793,* SRI Network Information Center, Menlo Park, CA, September, 1981.

[14] Fred B. Schneider. Replication Management using the State-Machine Approach. *Distributed Systems second edition,* Addison-Wesley, 1993, 169-197.