社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2007－MBL－43 (19)
2007－ITS－31 (19)
2007／11／22

トランスポート層におけるモビリティに関する一考察

久保 健　横田 英俊
（株）KDDI 研究所

本論文では、ネットワーク上の機能に依存せず、通信を行うエンドホスト間のみでモビリティを実現する手法を提案する。提案手法はトランスポート層においてモビリティを提供するが、コネクションの概念を拡張することにより特定のトランスポート層プロトコルに限定せず、あらゆるプロトコルに適用できる。本論文ではさらに、提案手法の実装について述べ、実験システムによるハンドオフの実験結果を示し、その実現性を示す。

A Study on Mobility Support in Transport Layer

Takeshi Kubo　Hidetoshi Yokota
KDDI R&D Labs. Inc.

In this paper, we propose a method of achieving end-to-end mobility in the transport layer. By introducing a concept of association which extends that of connection, the proposed method can be applied to any transport layer protocol, such as TCP, UDP, etc. We also demonstrate an implementation design for application of the proposed method. Furthermore, we show the feasibility of the proposed method through an experiment using the implemented system.

1. はじめに

無線 LAN や 3G 携帯電話システム等の無線技術や FTTH、DSL 等々の固定網技術といった様々なネットワークアクセス技術の普及により、個人宅、オフィス、公共スペースなどでのネットワーク利用が広がり、利便性も高まっている。さらに、ネットワーク間のシームレスな移動技術が普及し環境が整備されることで、ユーザは固定網や無線網を自由に移動し、インターネットへアクセスする状況が促進されると考えられる。

例えば CDMA2000 システム [1]では、データ通信における端末移動性（以後、端末移動性を単にモビリティと呼ぶ）を提供するために Mobile IP [2]をベースとした技術を利用している。一方、固定網においては端末が移動することが想定されていないため、Mobile IP のような移動管理の仕組みが導入されていることはほとんどない。そのため、無線網（CDMA2000）と固定網間のモビリティを提供するには、ネットワークシステムの変更が必要となり、モビリティ実現への障壁は高いと考えられる。また、他の多くのモビリティプロトコルについても、ネットワーク全体が同一のモビリティプロトコルを採用しなければならないという点で、Mobile IP と同様の問題を持っており、ユーザが「あらゆるネットワークへ移動できる」環境を提供することは困難である。

このような問題を克服する手立ての一つとして、エンドホストでモビリティをサポートする手法がある。筆者らはこの手法の中で、トランスポート層でモビリティを実現する方法を採用し、すべてのトランスポート層プロトコル（TCP、UDP 等）で、また NAT（Network Address Translation）存在下でも動作するモビリティ提供手法を提案する。さらに、本論文では、本提案手法の実装に関しても述べ、多数の OS で採用されているソケット通信方式を拡張し、アプリケーションに影響を与えることなく実現できることを示す。

2. 関連研究

これまでにもモビリティを提供するプロトコルが多数提案されているが、それらを大別すると、ネットワーク上の他のノードを利用して実現する方法とエンドホストのみで実現する方法の 2 種類に分けられる。

Mobile IP [2]やMobile IPv6 [3]はトンネリングを利用した手法であり、ネットワーク上に配置されたHome Agent （HA）が移動端末（MN）の現在位置（またはForeign Agent）までパケットをカプセル化して転送することによってモビリティが実現される。しかし、通信相手（CN）からMNへパケットが配送される際にHAを経由する必要があるため、三角経路と呼ばれる冗長な経路を生じる。MNはネットワーク移動のたびにHAへ位置登録を行うが、経路最適化[3, 4]と呼ばれる手法を用いれば、MNは通信中のCNへも直接位置登録を実施するため、三角経路の問題は解決されるが、依然としてHAを必要とする。

モビリティに関する議論では、locatorとidentifierの分離について取り上げられることが多い。例えばIPアドレスはホストの識別子（identifier）と場所を示す識別子（locator）を兼ねているといえが、このような二重性がモビリティ実現を困難にしている。一方、HIP [5]や*i3* [6]のようなオーバーレイネットワークでは、独自の識別子を導入することで、IPアドレスからidentifierの役割を切り離している。例えばHIPでは、Host Identifier（HI）が導入され（各ホストは自身の公開鍵をHIとして用いる）、各ホストはDNSサーバにこのHIを登録する。あるホストがパケットを送る場合、宛先となるホスト名（FQDN）からHIを解決し、さらにそのHIからIPアドレスを解決する。HIP上でのモビリティについてはrendezvous server [7]がホストのIPアドレス（locator）とHI（identifier）を逐一管理することによって、ホストが移動してもHIからIPアドレスへの解決ができるようになる。*i3*でも、ホストのidentifierとしてビット列が用いられ、IPアドレスはlocatorとして働く。Identifierとlocatorのマッピングについてのアルゴリズムや、シグナリングなど詳細な点は異なるものの、identifierからIPアドレスを解決するという処理の流れは*i3*もHIPと同様である。また、これらのプロトコルは、Mobile IPの経路最適化と同等の機能をもつ。しかし、新たな名前空間を定義し、名前解決手法を提供しなければならず、また当然ながらオーバーレイネットワークが事前に構築されていなければモビリティも提供できない。

SCTP [8] はコネクション型のトランスポート層プロトコルであり、これを拡張することによってモビリティを実現する手法が提案されている[9]。この手法はエンドホスト間のみでモビリティを実現するもので、ホストが新たなIPアドレスを取得した際に、通信相手ホストに新アドレスを通知することにより、相手ホストとの通信を継続できる。しかし、TCPやUDPに比べると、現時点ではSCTPはあまり普及しているとはいえない。

TCPを拡張することによってモビリティを実現するという提案[10, 11]もなされている。これらの方式の基本概念は[9]と同様であり、ホストが新たなアドレスを取得した際に、TCPオプションを拡張してそれを通知する。[10]では、認証鍵（公開鍵）をTCPの3-way handshakeに重畳して通知し、後のアドレス更新通知を受けた際にその正当性を当該認証鍵を用いて確認する。[11]では、3-way handshakeにおいてホスト間で暗号鍵の交換を行う。ホストが通信中に新たなアドレスを取得した場合、SYNパケット（拡張されたMigrate optionを付加する）を通信相手に送る。このMigrate optionには、通信開始時に交換された暗号鍵を用いて生成されたトークンが格納されており、受信者は通信相手（Migrate option送信者）の正当性を確認する。両手法ともこのような正当性確認を経て、受信者がコネクション情報を更新し、通信が継続される。しかしこれらの手法では、正当性確認のための情報に、IPアドレスやポート番号の情報が含まれていないため、コネクションが不正な攻撃にさらされる可能性がある。

本論文で筆者らが提案する方式は、[9, 10, 11]の提案と同様のアプローチをとるが、さらにあらゆるトランスポート層プロトコルに対応させるとともに、コネクションへの不正な攻撃を防ぐ手立ても備える。

3. プロトコル設計

トランスポート層（Layer 4）は、プロセス間の仮想的な通信チャネルを提供するという役割を果たす。この層にはコネクション型（TCPやSCTP等）とコネクションレス型（UDPやDCCP等）の2種類の方式がある。コネクション型プロトコルは、コネクションの確立、終了や再送制御などを提供するプロトコルであり、上位層に対して信頼性の高い通信を提供する。一方コネクションレス型プロトコルは、単純な機能のみ（例えば再送制

御無しなど）を上位層へ提供するため、コネクション型プロトコルと比べて処理のオーバヘッドが少ない。

本提案では、コネクション型、コネクションレス型に関係なく、事前に通信する2者間で association を確立しておき、この association に対してモビリティを提供する。ここで association とは、コネクションの概念を拡張した本提案独自のものであり、association を確立する通信相手と暗号鍵や識別番号などを交換する。なお、本論文ではコネクションは従来トランスポート層で取り扱われていたコネクションを指す。本提案手法では、ホストが新たな IP アドレスを取得した際に、association を確立している全相手に対してアドレス更新通知を行うことで、アドレス変更後も通信を継続させる。その際に association 確立時に交換した暗号鍵などを用いて正当性の確認を行う。

3.1. セキュリティに関する問題

上述のとおり、アドレス変化時にコネクションが更新されるが、成りすまし、再送攻撃、man-in-the-middle 攻撃などの不正によって、ノードの移動に乗じて通信相手が不正にすり替えられる可能性がある。したがって、不正なアドレス更新通知を適切に破棄することで、このような攻撃を防がなければならない。

一方で現在のインターネットにおいては、NAT および NAPT（Network Address Port Translation）の存在が無視できない（なお、本論文ではこれらを総称して NAT と呼ぶ）。しかし、トランスポート層を用いた[10]等のアプローチでは、アドレス更新通知が NAT を通過してパケットヘッダを書き換えられると、受信者（通信相手）がその変更が NAT によるものなのか攻撃によるものなのかを区別できないという問題が発生する。そこで、本提案ではこの問題も解決する。

3.2. 基本動作

本提案手法のモビリティ実現の流れは次のとおりである。(a) 通信開始時に 2 者間で association を確立する。(b) 一方がアドレスを更新した際、アドレス更新通知を通信相手に送信する。(c) ホスト間のパス上に NAT の存在を検知した場合には、正当性確認手順を実施する。(d) アドレス更新通知に基づき、コネクション情報を更新する。モビリティを直接的に実現するのは上記手順(d)であるが、これを適切に制御するために association が機能する。これらの手順を実施するために、表 1 に示す 6 種類の新たなシグナリングパケットを規定する。また、図 1 にこれらのシグナリングを用いた提案手法のシーケンスを、図 2 にはシグナリングメッセージの基本パケットフォーマットを示す。表 1 の 6 種類すべてのシグナリングに、initiator association ID、responder association ID、メッセージ認証子が基本情報として含まれる（各情報の詳細は後述する）。なお、メッセージ認証子の

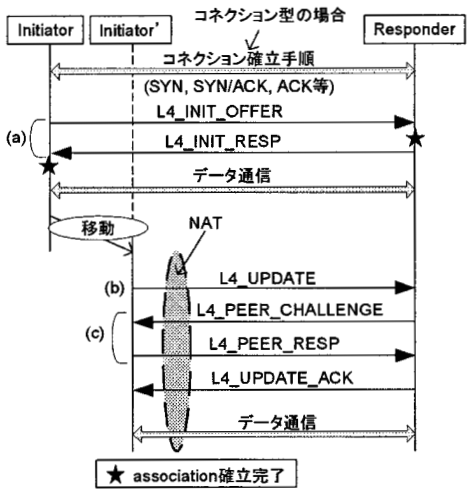

図 1　シグナリングシーケンス

表 1 シグナリングの種類

シグナリング種別	説明
L4_INIT_OFFER	Association 確立のためのメッセージ 共有暗号鍵および association ID を交換する。
L4_INIT_RESP	
L4_UPDATE	アドレス更新通知およびその応答
L4_UPDATE_ACK	
L4_PEER_CHALLENGE	NAT がパス上に存在する際の正当性確認 パケットヘッダとの不整合検出時に用いられる。
L4_PEER_RESP	

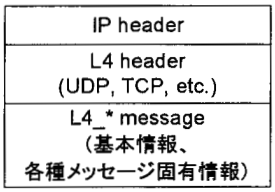

図2　パケットフォーマット

計算範囲は、図2の「L4_* message」全体であり、計算時には認証子部分は 0（ヌル）として認証子の値を算出する。

3.3. association の確立

表 2 にエンドホスト間で確立される association に含まれる情報を示す。なお、この association 情報はコネクション情報と同様に、両ホストにおいて同じ内容のものが保持される。本論文では、通信を初めに開始しようとしたノードを Initiator、通信を受け入れたノードを Responder として区別しているが、通信開始後は両者対等である。

図 1 に示すように、Initiator が L4_INIT_OFFER を送出する前に、トランスポート層のコネクション（例えば TCP コネクション）を先に確立する。これは、association がコネクションに完全に紐付けられたものであるためである。しかし、UDP のようなコネクションレス型のプロトコルの場合にはコネクション確立手順が存在しないため、先に association を確立した後に内部的にコネクションを生成する（詳細は 4.2 節で述べる）。

association 確立手順（図 1 の a）の主な役割は、共有暗号鍵および association ID の交換である。鍵交換には、Diffie-Hellman (DH) アルゴリズム[12]を用い、DH アルゴリズムで必要になる要素は、L4_INIT_OFFER および L4_INIT_RESP に格納される。なお、鍵交換のアルゴリズムは他の手法（例えば[11]で用いられる ECDH）を用いてもよい。ここで交換された共有暗号鍵は、以後のシグナリングメッセージのメッセージ認証子計算（例えば HMAC-SHA-1）等に用いられる。

association ID は、association を特定するための情報で、ホスト内で一意になるようにする。まず Initiator が L4_INIT_OFFER の initiator association ID に自ホスト内で一意となる値を、responder association ID には 0 を格納する。L4_ INIT_OFFER を受信した Responder はシグナリングメッセージ内の initiator association ID を association 情報に登録するとともに、自ホスト内で一意となる値を決めて responder association ID として L4_INIT_RESP に格納する。以後のシグナリングメッセージには、これら二つの ID を格納し、受信ホストで association の識別に用いる。なお、コネクションを識別するための情報（IP アドレスやポート番号）は、ホストの移動によって変化する可能性があるが、association ID は当該コネクションが存続する限り変化しない。

Initiator が L4_INIT_OFFER 送信後一定時間内に L4_INIT_RESP を受信しなかった場合、Initiator は L4_INIT_OFFER を再送する。また再送回数が上限に達した場合、association 確立失敗として OS に通知し、コネクションも終了させる。

3.4. アドレス更新通知

自身の IP アドレスが変化し、それを検知すると、該当するコネクション情報を書き換えるとともに、L4_UPDATE を通信相手に送信する。L4_UPDATE には、変更後の IP アドレス、宛先・送信元ポート番号、Initiator sequence No、Responder sequence No が含まれる。当シグナリングを受信した通信相手は、シーケンス番号とメッセージ認証子を確認し、当該メッセージを受け入れるかを判断

表 2　association 情報の内容

パラメータ	説明
Flag	Initiator か responder かを示す
Initiator association ID	Initiator ノード上での association 識別子
Responder association ID	Responder ノード上での association 識別子
Initiator sequence No.	再送攻撃を防ぐための情報。Initiator、Responder ノードがアドレス更新をするたびに増加させる。
Responder sequence No.	
Shared secret key	Association の正当性確認に用いられる共有暗号鍵
Challenge value	L4_PEER_CHALLENGE の Challenge 記憶用

する。正当なメッセージであれば、当該 association に対応するコネクション情報を書き換える。なお、コネクション情報の書き換えとは具体的にはコネクションの宛先 IP アドレスやポート番号を変更することを意味する。

L4_UPDATE の正当性確認に用いられるシーケンス番号は新規に L4_UPDATE を発行するたびに自身のシーケンス番号を 1 ずつ増加させる。例えば Initiator ホストが IP アドレスを変更して L4_UPDATE を送る場合、それまでの Initiator sequence No を 1 増加させる。受信側（Responder）では、そのシーケンス番号を記憶しておくことにより、再送攻撃（古いシーケンス番号で送られてくるシグナリング）を破棄することができる。

3.5. NAT 存在時の動作

L4_UPDATE メッセージの正当性確認ではメッセージ認証子によるメッセージ改ざんの検知も行われる。その際、ホスト間のパス上に NAT が存在すると、パケットの IP ヘッダ（L3 ヘッダ）や Layer 4 のポート番号（L4 ヘッダ）が書き換えられる。この書き換えにより、L4_UPDATE のメッセージ部に含まれるアドレスやポート番号と L3、L4 のヘッダの情報との不整合が生じる。しかしこの不整合は、悪意のある第三者によるパケット改ざん（man-in-the-middle 攻撃など）によっても発生する可能性があるため、この不整合の原因が NAT によるものかどうかを確認しなければならない。そこで本手法では、L4_PEER_CHALLENGE および L4_PEER_RESP メッセージを用い（図 1 の c）、通信相手が本当に L4_UPDATE を送信したのかどうかを確認する。

L4_UPDATE 受信したホストが、パケットヘッダとメッセージ部に不整合を検出すると、ランダムな値を Challenge value として L4_PEER_CHALLENGE を返答する。なお、L4_PEER_CHALLENGE を送る際の L3、L4 ヘッダは、受信した L4_UPDATE のヘッダを基に構成する。そして、L4_PEER_CHALLENGE を受信したホストは、Challenge value に対して、Response value を計算し、それを L4_PEER_RESP に格納して返答する。ここで、Response value は Challenge value と association 確立時に交換した共有秘密鍵を結合してハッシュをかけたものとする。L4_PEER_RESP を受信したホスト（つまり L4_UPDATE を受信したホスト）は、上記と同様の計算をして Response value が正しいかを確認する。もし、通信相手が正しく L4_PEER_RESP を返していれば、Response value も正しいはずであり、つまり L4_UPDATE もその通信相手から送られてきたと判断できる。その後、L4_UPDATE_ACK を返答して、アドレス更新処理が完了する。以上のようにして、L4_UPDATE の送信者を確認することにより、第三者による攻撃を検知し、不適切なシグナリングを破棄することができる。

4. 実装

4.1. 設計方針

Berkley Socket はプロセスが他のプロセスとネットワーク越しに通信を行うための仕組みとして、最も広く利用されている。例えば TCP では、プロセスは(i)ソケットを作成し(socket 関数)、(ii)ソケットにアドレスやポート情報を結びつけ(bind 関数)、(iii)相手ソケットとコネクションを確立し(connect, listen, accept 関数)、(iv)ソケットを通してパケットを送受信する(send, recv 関数等)。また UDP では (iii)は存在しないが、プロセスは connect 関数を呼ぶことで、相手ソケットとは無関係に自ソケットの中だけでコネクションが確立された状態を作ることも可能である。プロセスはこのコネクション（直接的にはその両端にあるソケット）を利用することで、パケットごとに送受信相手の設定や確認を行うことなくデータ送受信が可能となる。

パケットがどのコネクションに属するものかを示すための識別子として、宛先・送信元 IP アドレスおよび宛先・送信元ポート番号の 4 つ組の値とプロトコル種別（UDP や TCP）が用いられる（本論文ではこれらをまとめてソケット情報と呼ぶ）。そのため、通信中のノードの IP アドレスが移動などにより変化すると、受信パケットと該当ソケット情報の IP アドレスに関する情報に不整合が起こり、コネクションが無効になる(通信が切断される)。そこで本提案では、コネクション確立（上述のステップ iii）を実施した通信に対して、コネクション情報の更新を可能としモビリティを提供する。また、ソケット API を変更することなくこれを実現することにより、既存の

アプリケーションへの影響を避ける。なお、本論文ではLinux kernel-2.6.14を用いて提案手法の実装方法を述べる。

本実装では、図2のパケットをシグナリングメッセージ用の専用コネクション（ポート番号）を設けるのではなく、データ通信用のコネクションと同一のコネクションに重畳させる手法を採用した。これは、NATに特別なフィルタリングルール等を施さなくてもよいためである。

4.2. システムアーキテクチャ

図3に本提案手法を実装したシステムの概要を示す。図3のmobility部が本手法の中核をなし、シグナリング処理、associationの管理、ソケット情報（コネクション情報）の変更などを行う。また、図3のassociation部はassociation情報を格納したテーブルである。さらに、TCPおよびUDP処理部を拡張し、mobility部と連携させる。

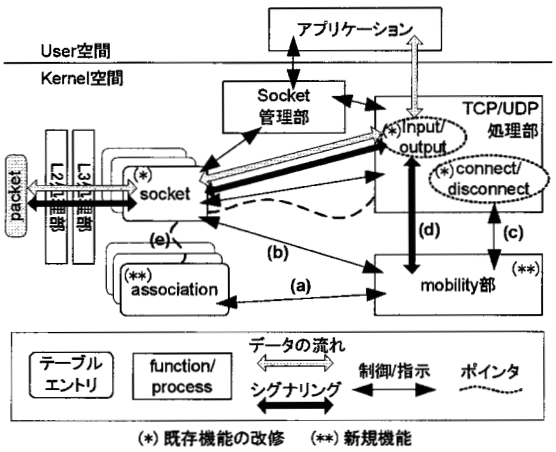

図3　システム概要

図3の(a)はassociation情報の制御を示しており、association情報の生成や破棄、情報へアクセスを表している。図3の(b)はソケット情報の更新であり、ネットワークインタフェースのIPアドレスが変化したことを検知したときや、有効なL4_UPDATEを受信したときに実施される。図3の(c)はmobility部とLayer 4のコネクション確立処理部の間の連携を示しており、TCPの場合にはコネクション確立後にassociation確立を指示する。一方UDPの場合には、Initiatorではconnect関数が呼ばれたことを契機にassociation確立を指示するが、逆にResponderではL4_INIT_OFFER受信時に対応するコネクションが存在しないため、mobility部からコネクション生成の指示を出す。図3の(d)は、本手法のすべてのシグナリングメッセージをmobility部に引き込む、またはmobility部から出力する流れを示している。つまりTCPやUDPの入力関数に、本メッセージを振り分ける機能を追加する。なお、本実装では、TCPにはmobilityオプションを追加して振り分けを可能とする。またUDPでは、UDPのチェックサムを0とし、UDPヘッダの直後8バイトにマジックナンバー（事前に定義しておく）を埋め込むことによって、振り分けを可能とした。最後に図3の(e)はassociationとソケット（コネクション）の関連付け（ポインタ）を示している。

移動（アドレス変更）を検知するための仕組みとして、本実装ではLinuxカーネルが提供する"notifier"メカニズムを利用する。このnotifierを使えば、ネットワークインタフェースのIPアドレスの変化を監視するようにカーネルに登録しておくことによって、該当のイベントが起こった際に、通知を受けることができる。この通知を契機として、mobility部が3.4節で述べたアドレス更新通知手順を開始する。

図3に示すように、アプリケーション層はmobility部やassociationについて関知しない。したがって、ソケットAPIを変更することなく本手法を実現できる。ただし、より柔軟な制御（例えば本手法の有効/無効を切り替えるなど）を可能にするために、例えばsetsockopt関数などを用いてmobility部へのアクセスを提供してもよい。

4.3. 考察

前節では本提案手法を、TCPおよびUDPコネクションに対して実現するための実装方式を示した。これと同様の方針で、あらゆるトランスポート層のプロトコルに本手法が適用可能である。しかし、アプリケーションの観点からは、次のような制約があると考えられる。

1) UDP通信でconnect関数を呼ぶアプリケーションはあまりない。

→ モビリティが必要となるのは、VoIP のような比較的長時間通信を持続するタイプのアプリケーションであり、これに関しては connect 関数を呼ぶようにだけプログラムを変更する必要がある。しかし、DNS query/ response のような短期間で終了するようなアプリケーションには、そもそもモビリティ自体が不要であると考えられるためアプリケーションの改修は不要である。

2) アプリケーション自体が通信相手の IP アドレスを保持する場合、本手法を適用するとアプリケーションが正しく動作しなくなる可能性がある。

→ 例えば FTP クライアントでは、ユーザがファイル転送のリクエストを発行するたびに bind 関数、connect 関数を呼ぶ。その際、アプリケーション自体が FTP サーバの IP アドレスを保持しているため、クライアントが IP アドレスを変更した後に新たなリクエストを発行しようとするとエラーが発生する ("cannot assign requested address" または "host unreachable")。これを避けるには、bind 関数、connect 関数を呼ぶ前に、getpeername 関数などでソケット情報から IP アドレスを取得するようにすればよい。なお、通信中のファイル転送はこのような変更なしでも移動時に通信が継続される。

3) 両ホストが同時に移動すると、お互いが通信相手を見失ってしまう。

→ 両者が同時移動すると、どちらのアドレス更新通知も通信相手に到達しないため通信が切断してしまう。この問題は、本手法のみでは解決できず、別の仕組みの助けを要する。例えば Dynamic DNS [13]の DNS Update を利用する方法がある。つまり、移動ホストはアドレス更新時に直ちに DNS Update を用いて、自ホスト名と IP アドレスのマッピングを更新する。そして、L4_UPDATE 送信後、一定時間内に L4_UPDATE_ACK を受信しなければ、同時移動が発生した可能性があるため、DNS query によって通信相手の IP アドレスを解決し、再度 L4_UPDATE を送信する（ただし、association 確立時にホスト名を交換しておく必要がある）。これによって同時移動の問題も解決できると考えられる。

4.4. 実験

筆者らは、本提案手法を実装し、図 4 に示すような実験ネットワークでハンドオフ実験を行った。実験ネットワークは 3 つのネットワークから構成されており、ルータは NetB に対して NAT サービスを提供している。また、ネットワークはすべて 100base-TX で接続されている。NetA および NetB ではユーザは DHCP によってアドレスを取得する。なお、実験ではクライアントは移動直後に手動で DHCP クライアント（dhclient）を起動して新規アドレスを取得した。

実験では、TCP および UDP の通信をしながら NetA から NetB へ移動した。TCP 通信として FTP を用いてサーバからのファイルダウンロードを行い、UDP 通信としては、128 バイトのデータパケットをサーバから受信した（この UDP アプリケーションではクライアント、サーバ双方で connect 関数を呼んでいる）。なお、図 4 に示すように、移動ユーザが Initiator、サーバが Responder となる。

図 5 および図 6 に、UDP および TCP 通信時の実験結果を示す。両図の横軸は経過時間、縦軸は移動ノードへ正しく到達したパケットの累積数を示している。図 5 および図 6 では、ネットワーク間を物理的に移動した後、DHCP シーケンスが開始されるまでの時間が 3 秒程度と長くなっている。これは、ネットワーク間の物理的な移動 (LAN ケーブルの差し替えを行っているが、これ自体は数 10ms 程度で完了している）の後、dhclient を起動した際に、実際の DHCP シーケンスが開始されるまでに時間がかかっているためであり、dhclient の実装依存だと考えられる。DHCP シーケンスが完了し、新たな IP アドレスを取得した後、すぐに L4_UPDATE、L4_PEER_CHALLENGE、L4_PEER_RESP、L4_UPDATE_ACK が授受され、データ通信が復旧している。なお、図 5 の UDP では、L4_UPDATE_ACK 受信後すぐにデータ通信が復旧しているが、図 6 の TCP では、L4_UPDATE_ACK 受信から約 0.6 秒のタイ

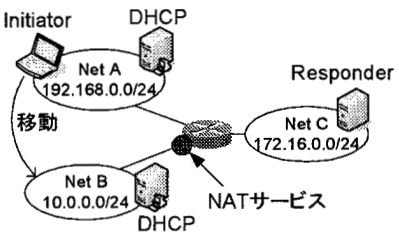

図 4 実験ネットワーク

ムラグがある。これは、TCP の再送タイマによるものである（本実装では、TCP 再送タイマには手を加えていないため、L4_UPDATE_ACK と連動していない）。

本節では、提案手法を実現するためのシステムアーキテクチャについて述べ、さらに実装システムを用いたハンドオフの実験結果を示した。

5. 結論

本論文では、あらゆるトランスポート層プロトコルでモビリティ実現し、また NAT の存在下でも動作する手法を提案した。本手法では、コネクションの概念を拡張した association を導入し、この association に基づいてコネクション情報の変更を制御することによってモビリティを実現している。

さらに、提案手法の実装についても述べ、ソケット API には変更を加える必要がないため、既存のアプリケーションを変更することなく、または大きな改修を加えることなくモビリティを享受できることを示した。それ故に、これから生み出されるアプリケーションは容易に本提案手法を利用することができると考えられる。

日ごろご指導いただく、KDDI 研究所秋葉所長に深く感謝いたします。

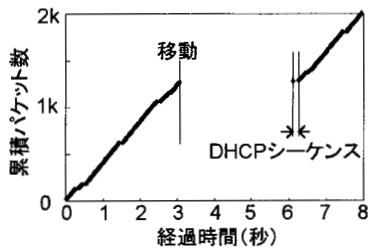

図 5 UDP ダウンロード

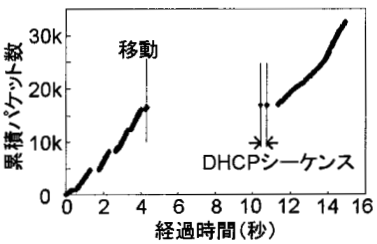

図 6 TCP ダウンロード

参考文献

[1] Wireless IP Network Standard, 3GPP2 P.S0001-A version 3.0.0, 2001.

[2] C. Perkins, IP Mobility Support for IPv4, RFC3344, IETF, 2002.

[3] D. Johnson, C. Perkins, and J. Arkko, Mobility Support in IPv6, RFC3775, IETF, 2004.

[4] C. Perkins and D. Johnson, Route Optimization in Mobile IP, draft-ietf-mobileip-optim-11, IETF, 2001.

[5] R. Moskowitz, P. Nikander, P. Jokela and T. Henderson, Host Identity Protocol, draft-ietf-hip-base-07, internet draft, IETF, 2007.

[6] Ion Stoica, Daniel Adkins, Shelley Zhuang, Scott Shenker, Sonesh Surana, Internet Indirection Infrastructure, Proceedings of ACM SIGCOMM, 2002.

[7] J. Laganier and L. Eggert, Host Identity Protocol (HIP) Rendezvous Extension, draft-ietf-hip-rvs-05, internet draft, IETF, 2006.

[8] Q. Xie, et. al., Stream Control Transmission Protocol, RFC2960, IETF, 2000.

[9] R. Stewart, Q. Xie, M. Tuexen, S. Maruyama and M. Kozuka, Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration, draft-ietf-tsvwg-addip-sctp-18, internet draft, IETF, 2007.

[10] Daichi Funato, Kinuko Yasuda and Hideo Tokuda, TCP-R: TCP Mobility Support for Continuous Operation, Proceedings of IEEE ICNP, 1997.

[11] Alex C. Snoeren and H. Balakrishnan., An End-to-End Approach to Host Mobility, Proceedings of ACM MOBICOM, 2000.

[12] W. Diffie and M. E. Hellman, New Directions in Cryptography, IEEE Transactions on Information Theory, vol.IT-22, No.6, pp.644-654, 1976.

[13] P. Vixie, et. al., Dynamic Updates in the Domain Name System (DNS UPDATE), RFC 2136, IETF, 1997.