L_006

トランスポート層におけるモビリティサポートに関する提案

A Proposal of Mobility Support in Transport Layer

久保 健† 横田 英俊† 井戸上 彰†
Takeshi Kubo Hidetoshi Yokota Akira Idoue

1．まえがき

近年、ユーザがあらゆる場所でインターネットを利用できる環境が整いつつある。これによりユーザが通信しながら自由に移動するという利用形態が可能となり、このような利用形態を実現するため、モビリティに関する研究開発が盛んに行われている。

モビリティ実現手法としては、MIP[1], MIPv6[2]のようなネットワーク層での tunneling 技術を利用した手法や、HIP[3]、I3[4]のような overlay 技術を利用する手法等がある。これらの手法では、何らかのサーバがノードの ID と現在位置をマッピングし、必要に応じてパケットを移動先へ転送する。このようにノード間の通信にサーバが直接介在するため、これらの手法を実現するためには、ノードへの機能追加だけでなくネットワークへのサーバ追加などの対応が必要となる。これらとは別に、トランスポート層やセッション層でモビリティを実現する手法[5-8]等も提案されている。これらの手法は、End-to-End でモビリティの管理を行うため、コネクション確立後の移動にはネットワーク内のサーバを必要としない。しかし、[5-7]は特定のトランスポートプロトコルに限定するものであり、[8]は専用のライブラリや関数を必要とする。

本論文では、多くの Operating System (OS)に実装されているソケット通信の仕組みを拡張し、ネットワーク内にサーバを必要とせず、またトランスポート層のプロトコルに拠らず、モビリティを実現する手法を提案する。

2．ソケット通信

トランスポート層（L4）の主な役割は、2 つのプロセス間で仮想的な通信路を実現することである。TCP のようなコネクション型プロトコルは、コネクション確立・終了手順や受信確認、再送制御などの機構を持ち、上位層へ高信頼な通信を提供する。逆に UDP のようなコネクションレス型プロトコルは、上記のような機構を持たず信頼性を保証しない代わりにオーバーヘッドの少ない通信を提供する。

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、Berkley Socket が最も広く利用されている。例えば TCP では、プロセスは(1)ソケットを作成し(socket 関数)、(2)ソケットにアドレスやポート情報を結びつけ(bind 関数)、(3)相手ソケットとコネクションを確立し(connect, listen, accept 関数)、(4)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 ではコネクション確立手順が無いため(3)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（コネクト状態）を作ることも可能である。つまりコネクト状態とは、ソケットに結び付けられた通信相手ソケットの存在（相手の IP アドレスやポート番号）に関しても OS が把握している状態であり、プロセスはこのソケットを通すことによって、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。

† （株）KDDI研究所

3．提案方式

通常のソケット通信では、パケットを受信した OS は送信元 IP アドレス、宛先 IP アドレス、送信元ポート、宛先ポートの 4 つの値の組とプロトコル種別（UDP、TCP 等）によって、受信パケットがどのソケットに対応するかを識別し、対応するプロセスへ受信データを渡す。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、(a)物理的に相手からのパケットが届かなくなる、またたとえパケットを受け取れたとしても、(b)受信パケットと該当ソケットの 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）。

モビリティ実現には上記(a)、(b)の問題解決が必要であるが、本提案手法ではそれぞれ次の(A)、(B)の方針でこれを達成する。(A)自ノードの IP アドレスが変化したときには、正しい通信相手ソケットにそれを通知し、新しい宛先へパケットが送信されるようにする。(B)パケットを受信した OS は正しい通信相手ソケットからの受信であるかどうかという点を基に、対応するソケットを識別する。ここで、「正しい通信相手ソケット」とは、一度確立されたコネクションの相手ソケットが途中で不正に変更されず、2 つのソケットが一貫していることを指す。

コネクションレス型プロトコルである UDP は、DNS や各種シグナリングなどのように少ないやり取りで完結する通信に多く用いられる。また、その通信効率の良さから VoIP や動画ストリーミングのようなリアルタイムアプリケーションにも利用される。本提案手法では、後者のように比較的長時間にわたって同一の通信相手と通信を続ける場合についても、コネクト状態を作ることによってモビリティを提供可能とする。

3.1 メッセージ認証による通信相手の確認

ノードの移動などによる状態変化（アドレス変化）の前後のコネクション維持では、上述(A)、(B)に示すように、お互いが「正しい通信相手ソケット」であるかどうかが重要である。本提案手法では、受信パケットが正しい通信相手ソケットから不正な改ざん無く届いたものであるかどうかをメッセージ認証コード（MAC）によって確認し、ソケットの識別およびプロセスへの受信データ引渡しの指標とする。具体的には、ソケット間でコネクションを確立する際に、Diffie-Hellman 鍵交換アルゴリズム[9]によってソケット間で共通暗号鍵を交換し、その後のパケットにはその鍵を用いた HMAC-MD5 等を付加する（計算範囲は図 1）。受信側は該当ソケットが保持している鍵を用いて HMAC-MD5 を計算し、パケットに付加された値と同じであれば受信データをプロセスへ渡し、異なれば破棄する。

3.2 コネクションの状態管理

提案手法では、上記(A)において明示的なシグナリングを送らず受信パケットの送信元アドレスが変化していれば対応するソケットの情報（通信相手のアドレス）を更新する。ただし、移動直後に送信すべきパケットが無い場合に

は、空のデータを送るなどしてノードの移動を通信相手に素早く伝えなければならない（このようなパケットを明示的なシグナリングとして定義してもよい）。また、不正なノードによる replay 攻撃を防止するために、ソケットの状態に関するシーケンス番号を導入し各パケットに付加する。コネクト状態にあるソケットは、自ソケットおよび相手ソケットのシーケンスを管理しておき、保持している相手シーケンスより大きな値を持つパケットを受信したときのみ、ソケット情報を更新する（もちろん 3.1 節の MAC の確認が前提である）。なお、自分自身のシーケンスは、IP アドレスの変化を契機に増加させる。

3.3 ポート番号について

本提案手法において OS のソケット識別の最も重要な指標は MAC であるが、それに加えて従来どおり、宛先ポート、送信元ポート（ポート番号対）も利用する。これはソケット識別の際の検索効率を向上するための目印や、コネクション確立前の待ち受けポート、また通常の L4 のパケットフォーマットとの整合性を保つという役割を果たす。

従来手法は、複数のコネクションで同一ポート番号対を用いても相手のアドレスが違えば OS 上でソケットを一意に指定できたが、本手法では IP アドレスを考慮しないため、コネクション確立時にポート番号対が互いのノード上で一意になるように折衝が必要となる。

3.4 コネクション確立および移動手順

本提案におけるコネクション確立では、3.1 節や 3.3 節に示した通り共通暗号鍵の交換やポート番号の折衝が必要となるため、L4_INIT および L4_OFFER という新たなシグナリングを導入する。これらのシグナリングデータ、前節までの MAC やシーケンスは、図 1 に示す拡張 L4 ヘッダに格納する。図 2、3 に本提案のコネクション確立手順を示す。両図に示す L4_INIT および L4_OFFER によって Diffie-Hellman 鍵交換アルゴリズムと使用ポート策定のためのネゴシエーションを行う。図 3 では、3-way handshake に重畳することで余分なパケット交換を削減している。なお、使用ポート番号策定では、L4_INIT において候補となるポート番号対を提案し、L4_OFFER でどれを使用するかを決定する。万一決定できなかった場合には、L4_OFFER で可能性のあるポートを通知し、さらに再度ノード A から L4_INIT で新たなポートを提案し、決定するまで繰り返す。

図 4 に、ノード A が別ネットワークへ移動した際の手順を示す（手順は、UDP、TCP とも同様である）。ノード A は新アドレス取得後（図 4 の A'）シーケンスを 1 増加させてデータパケットを送信し、ノード B はソケット情報を更新し次回以降のパケットを移動後のノード A 宛に送信する。

4．実装と評価

筆者らは、OS に Linux-2.6.14 を用いて提案手法を実装した。本実装では、プロセスが connect 関数を呼んだ際に L4_INIT を付加するかどうかを ioctl を用いて制御可能であり、L4_INIT を付加しない場合には従来どおりのソケット通信を行う。図 5 に実験環境と実験手順を示す。なお、FTP アプリケーションには全く改修を加えていない。

提案手法および従来手法における TCP の 3-way handshake 完了までにかかる時間は、それぞれ 2.3ms、0.18ms となった。この違いの要因は Diffie-Hellman アルゴリズムでの鍵生成処理であると考えられる。また、提案手法でのネットワーク移動からデータ転送再開までの時間は 12.8s であった。これは、TCP の retransmission のタイミングで転送が再開されたためである。

5．まとめ

本稿では、コネクト状態にあるソケット間の通信において、拡張 L4 ヘッダに付加された MAC や状態シーケンスを利用することにより、トランスポート層における End-to-End のモビリティを提供する手法を提案し、実装結果を報告した。今後は、本方式での 2 ノード同時移動のサポートや NAT 越えに関する問題の検討を行う。

日ごろご指導いただく KDDI 研究所秋葉所長に感謝いたします。

参考文献

[1] C. Perkins, “IP Mobility Support for IPv4,” RFC3344, IETF, 2002.
[2] D. Johnson, et al., “Mobility Support in IPv6,” RFC3775, IETF, 2004.
[3] R. Moskowitz, et al., “Host Identity Protocol,” draft-ietf-hip-base-06, internet draft, IETF, 2006.
[4] Ion Stoica, et al., "Internet Indirection Infrastructure," Proc. ACM SIGCOMM'02, 2002.
[5] R. Stewart, et al., “Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration,” draft-ietf-tsvwg-addip-sctp-15, internet draft, IETF, 2006.
[6] Daichi Funato, et al., “TCP-R: TCP Mobility Support for Continuous Operation“, Proc. IEEE ICNP'97, 1997.
[7] Alex C. Snoeren, et al., “An End-to-End Approach to Host Mobility,” Proc. ACM MOBICOM'00, 2000.
[8] 金子晋丈他, “多様化するインターネット環境のためのセッションレイヤモビリティサポート,” 電子情報通信学会総合大会, B-7-14, 2002.
[9] E. Rescorla, “Diffie-Hellman Key Agreement Method,” RFC2631, IETF, 1999.

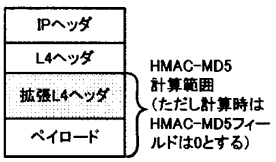

図1 パケットフォーマット

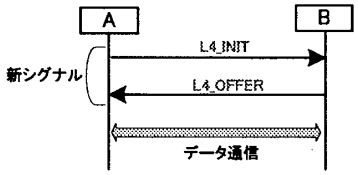

図2 コネクション確立手順(UDP)

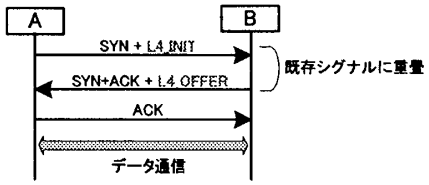

図3 コネクション確立手順(TCP)

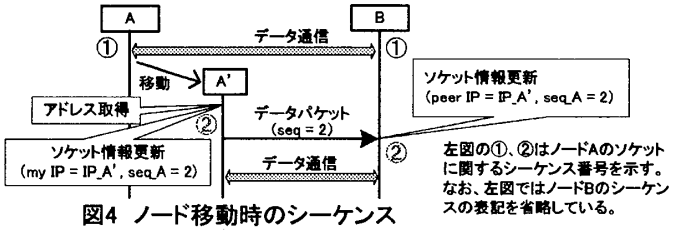

図4 ノード移動時のシーケンス

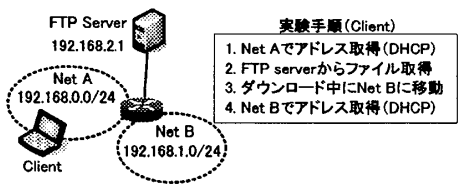

図5 実験ネットワークと実験手順