システムソフトウェアと
オペレーティング・システム 78－19
（１９９８．５．８）

移動型計算機における Disconnected Operationの設計と実装

永田 智大[1] 望月 祐洋[1] 徳田 英幸[1,2]
[1]慶應義塾大学政策・メディア研究科[2]慶應義塾大学環境情報学部

移動型計算機を取り囲むネットワーク環境はまだ十分に整備されていない．そのため，可搬性のある移動型計算機がネットワーク上を移動した際，様々な問題が生じる．移動型計算機がネットワークから長時間切断された場合，それまで生成していたコネクションがTCPのタイムアウトにより終了してしまう．移動型計算機がネットワークに再接続した際，再びコネクションを生成し直す必要がある．つまり，透過性が保証されていない．本研究では，移動型計算機が頻繁にネットワークを切断，再接続した際，それまで生成されていたコネクションの再利用を実現するMDO (Mobile-ip based Disconnected Operation) の設計と実装を行った．MDOでは，ネットワーク切断時に生じるTCPコネクションの切断を防ぎ，再接続時に保持されているTCPコネクションを継続して利用できる．

MDO: The Design and Implementation of Disconnected Operation for Mobile Computing Environment

Tomohiro Nagata[2] Masahiro Mochizuki[2] Hideyuki Tokuda[1]
[1]Graduate School of Media and Governance, Keio University
[2]Faculty of Environmental Information, Keio University

The network environment in relation to the mobile computer is not sufficiently equipped. When a relocatable mobile computer moves over the network, several problems occur. The mobile computer is disconnected after a given amount of time by TCP time out. It is necessary to reconnect the mobile computer on to the network and re-open the connection. Transparency is not guaranteed. In this research, we designed and implemented the Disconnected Operation - MDO (Mobile-ip based Disconnected Operation) -, enabling the user to reestablish the maintained TCP connection on physical disconnecting and reconnecting of the mobile computer.

1 はじめに

従来のインターネットでは，ファイルの転送などがその主たる利用方法であった．しかし，電子メールやWWW(World Wide Web)の急激な普及により，いつでもどこでも計算機からネットワークを利用する必要が出てきた．

人々の移動に伴って持ち運びの容易な計算機，つまり可搬性のある移動型計算機が多く利用される．以前の移動型計算機は物理的な制約のため，卓上型計算機よりもはるかに処理能力が劣っていた．しかし，様々なデバイスの小型化，性能向上によって，卓上型計算機と同じ環境を移動型計算機で利用できる．それに伴い，オペレーティングシステムも，移動型計算機がより利用しやすいサービスを提供するようになった．

このような移動型計算機の発達に対し，それを取り囲むネットワーク環境は，移動型計算機に適したサービスを提供していない．オペレーティングシステムと異なり，ネットワーク環境で新たなサービスを提供する場合，様々な機能を多くの計算機に実装する必要がある．そのため，現在のネットワーク環境は移動型計算機に適していない．

可搬性の高い移動型計算機がネットワークから切断，再接続した際，TCPのタイムアウトによってコネクションを切断される場合がある．そのため，移動型計算機がネットワークに再接続した際，

既に生成されていたコネクションがすでに切断され，コネクションを新たに生成し直す必要がある．つまり，TCP 層におけるコネクションの透過性が保証されていない．

このような問題に対し，MDO は TCP 層における透過性を保証するため，Mobile-IP と現在の TCP の問題を考慮して設計されている．Mobile-IP は IP 層における透過性のみを保証しており，MDO はこれを拡張して TCP 層における透過性を簡潔な方法で保証する．

また，MDO は現在の TCP との互換性を保証し，TCP のプロトコルをできるだけ変更しないように設計されている．そのため，移動型計算機が MDO を実装していない計算機と通信を行っても，通常の IP や TCP が提供するサービスをそのまま利用でき，MDO の提供するサービスも利用できる．

TCP のコネクションには様々な状態があり，移動型計算機がネットワークから切断された際，どのような状態でも TCP 層における透過性を保証できるように，MDO は設計されている．

2 TCP 層における透過性

TCP 層における透過性とは，計算機がネットワークから切断された場合でも，それまでのコネクションは切断されずに保持され，再び計算機がネットワークに接続した時点で，その保持されたコネクションを継続して利用できることを指す．つまり，ネットワークから長時間切断していても，コネクションの同一性を保持することをいう．

TCP ではコネクションを確立することにより，通信の信頼性を保証している．通信相手が通信不能になった場合に備えタイマを設定し，タイムアウトによってコネクションが切断されるように TCP は実装されている．

移動型計算機が長時間ネットワークから切断された場合，telnet などのアプリケーションが生成したコネクションは，TCP のタイムアウトにより切断されてしまう．そのため，移動型計算機が再びネットワークに接続した場合，切断する以前のコネクションを継続して使用できず，アプリケーションを再起動して，新たにコネクションを生成し直さなければならない．

そのため，現在の TCP では透過性が保証されていない．

2.1 再送タイマ

通常の TCP のコネクションでは，TCP セグメントが送られてくると相手は応答を送信しなければならない．もし応答を受信しなかった場合，一時的な途中経路の不調と考え，再送タイマによって再送を試みる．もし，ある一定時間送信を試みると相手が通信不能であると判断し，TCP はコネクションを切断する．再送タイマの状態遷移の様子を図 1に示す．

移動型計算機がネットワークから切断された場合，再送タイマの時間内に再接続すれば，コネクションは再送タイマによって切断されない．つまり，無線 LAN でローミングを行うために，一時的にネットワークから切断される場合などは問題ない．

しかし，移動型計算機が長距離の移動のために，長時間ネットワークから切断されている場合，それまで生成されていたコネクションは切断される．

そのため，移動型計算機がネットワークに再接続した際，アプリケーションを再起動してコネクションを再生成する必要がある．

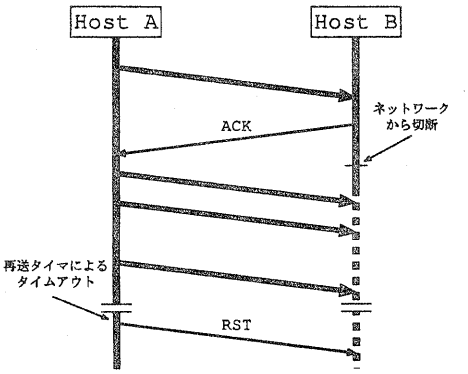

図 1: 再送タイマの状態遷移

2.2 キープアライブタイマ

移動型計算機がネットワークから切断されている間に，TCP セグメントが送信されなければ，再送タイマによって切断されることはない．しかし，この場合，キープアライブタイマによって切断されてる．

計算機が通信不能でない場合でも，TCP コネクション上にデータが一切送られないことがある．そのため，相手が通信可能かどうかを検出するため，キープアライブタイマによって，2 時間毎にキープアライブパケットを送信している．もしキープアライブパケットを送信したにもかかわらず，相手からの応答がない場合，TCP はコネクションを切断する．キープアライブタイマの状態遷移の様子を図 2に示す．

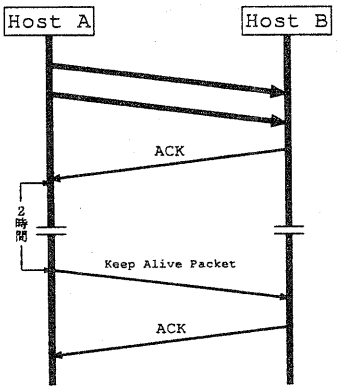

図 2: キープアライブタイマの状態遷移

移動型計算機がネットワークから切断されている間に，TCP セグメントが送信されなくても，長時間ネットワークから切断され続けていると，キープアライブタイマによって，それまで生成されていたコネクションが切断されてしまう．

3 関連研究

移動型計算機とネットワーク環境に関して，様々な研究が行われている．それらのうち，代表的なものを述べる．

I-TCP I-TCP[1] では，TCP を改良することにより，移動型計算機が無線 LAN を利用した際の通信効率を向上させる．
無線 LAN によるネットワーク間の移動が生じるため，I-TCP は IP 層での透過性を保証している．無線 LAN のローミングのため，移動型計算機がネットワークから切断される時間は短い．そのため，I-TCP は長時間移動型計算機がネットワークから切断されることを考慮していない．

Rover Rover[2] は queued remote procedure call(QRPC) と，relocatable dynamic objects(RDOs) を使用したツールキットを提供する．Rover を利用してアプリケーションを設計，実装することにより，移動型計算機による通信を円滑に行える．
しかし，Rover の提供するツールキットを利用するために，アプリケーションを変更する必要がある．そのため，移動型計算機は利用するアプリケーションに変更を加える必要がある．

Web Express ARTour Web Express[3] は，Proxy サーバを 2 つ用意する．1 つは Client Side Interceptor(CSI) で，移動型計算機のデバイス内で存在する．もう 1 つは Server side Interceptor(SSI) で，ネットワーク上に存在する．移動型計算機上の Web ブラウザからの要求は，CSI を通じて送信される．移動型計算機がネットワークから切断されている間，CSI がキャッシュを検索したり，キューとして保持しておく．
しかし，Web ブラウザ以外のアプリケーションでは Disconnected Operation のサービスを利用できない．

TCP-R TCP-R[4] は TCP の状態遷移に新たな状態を加え，移動型計算機がネットワークから切断されている状態では TCP のタイマを停止し，タイムアウトによるコネクションの切断を防ぐ．また，移動型計算機が異なるネットワーク上に移動した際，移動型計算機の新たな IP アドレスを取得し，TCP 層であて先 IP アドレスを変更する．
TCP-R を利用する場合，移動型計算機と通信相手の計算機の両方が TCP-R を実装する必要がある．そのため，TCP-R を利用するための環境を構築しにくい．

4 MDO の概要

RFC2006[7] は，Mobile-IP を利用することにより，トランスポート層などの上位層における透過性を保証するとしている．しかしこれは短時間の切断や，同一の IP アドレスの同一のポート番号で通信できることしか指していない．つまり，長時間ネットワークから切断された MN が再接続した際，切断以前のコネクションを再利用することについては言及されていない．

MDO では，MN(Mobile Node) がネットワークから切断されている間，HA(Home Agent) が MN に成り変って，それまでのコネクションを保持する．MN の通信相手である CH(Correspondent Host) から送信されてくる TCP セグメントに対し，MN に成り変った HA は CH に対して応答を送信する．また，HA 上の Proxy サーバは CH から送信されてくる TCP セグメントを，一時的に保持しておく．これによって TCP のタイムアウトによる，CH と MN 間のコネクションの切断を防ぐ．MDO における CH と HA の関係を図 3に示す．

MN が再びネットワークに接続した際，HA は CH に成り変わって，Proxy サーバに一時的に保持されていた TCP セグメントを MN に送信するため，移動先のネットワーク上の FA(Foreign Agent) に転送する．送信されてきた TCP セグメントを受け取った MN は，HA に対して応答を送信する．これにより，ネットワークから切断されている間

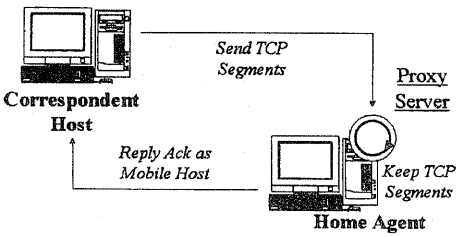

図 3: MDO における CH と HA の関係

に送信されてきた TCP セグメントを，MN は取得できる．MDO における HA と MN の関係を図 4 に示す．

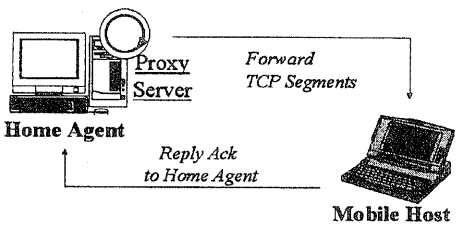

図 4: MDO における HA と MN の関係

このように Mobile-IP をベースプロトコルとして設計する際，Mobile-IP の各エージェントの拡張を行う必要がある．

5 エージェントの拡張機能

5.1 MN の状況把握

MDO では，現在どの MN がネットワークに接続されているのか，または切断されているのかを TCP 層で把握し，その状態に適した処理を行う必要がある．

また MN も，現在自分がネットワークに接続されているのか，されていないのか，状況を把握する必要がある．もし，ネットワークに接続されていない場合，自らの TCP のタイマを停止し，TCP コネクションを保持しなければならない．

そこで，MN は自分の状況を把握するために，Mobile-IP のレジストレーションを利用する．Mobile-IP では，MN がネットワークに接続した際，HA の UDP434 番ポートに対してレジストレーションを行う．レジストレーションが行われた場合，MN はネットワークに接続している．また，レジストレーションの有効期限が切れた場合，MN はネットワークから切断されている．MDO は，Mobile-IP のレジストレーションを利用して，MN の状態を把握する．

また MDO では，通常の TCP にできるだけ変更を加えないように設計しているため，HA と MN の拡張だけで MDO の提供するサービスを利用できる．しかし，これだけでは MN の通信相手である CH が，MN の状況を把握できない．

そこで，MDO ではオプションとして CH の機能を拡張し，現在 MN がネットワークから切断されているかどうかを把握する．

MDO では，エージェントとカーネルとの通信に，ルーティングソケットを利用する．図 5にルーティングソケットのコマンドを示す．

```
#define RTM_MIPADD      0x31
#define RTM_MIPDELETE   0x32
#define RTM_MIPCHANGE   0x33
#define RTM_MIPCLEAR    0x34
#define RTM_ARPOFF      0x35
```

図 5: ルーティングソケットのコマンド

ルーティングソケットを通じて MN の状況を通知されるカーネルは，MN の状況を dhl_entry 構造体を利用して保持する．図 6に dhl_entry 構造体を示す．

```
struct dhl_entry {
    LIST_ENTRY(dhl_entry) dhl_link;
    struct sockaddr dhl_addr;
                    /* home address */
    struct sockaddr dhl_dummy;
                    /* dummy entry */
    u_long dhl_flags;
              /* flags (below) */
              /* (host byte order) */
};
```

図 6: dhl_entry 構造体

この dhl_entry 構造体は，リスト構造としてカーネル内に保持されている．リスト内に MN の IP アドレスを含む dhl_entry 構造体がある場合，その MN はネットワークから切断されていることになる．

HA は常に MN の状況を把握している．もし MN がネットワークから切断された場合，HA は CH に対して MN の状態を通知する．HA からの通知を受けた CH はコンソールやダイアログなどを利用して，ユーザに通知できる．これにより，単なる経路の不調によって通信不能なのか，MN がネッ

トワークから切断されたために通信不能なのかを把握できる．

5.2 成り変わり

MNがネットワークから切断されている間も，コネクションが成立しているため，TCPセグメントは送信され続ける．送信されるTCPセグメントに対し応答を受信しないため，再送タイマやキープアライブタイマによって，コネクションが切断される．そこで，MHがネットワークから切断されている間，MNに送信されてくるTCPセグメントをHAは取得，保持し，応答を送信してコネクションを保持する必要がある．

通常，データリンク層では，IPアドレスとMACアドレスのARPテーブルによって，どのインターフェースにデータを送信するのかを決定する．このテーブルはARPを利用して登録，変更される．MDOでは，HAがARPテーブルを変更することによって，送信されてくるTCPセグメントをHAがMNに成り変って取得する．

TCP層では，IP層から送られてきたデータをTCPコントロールブロックを利用して，管理をしている．通常，TCPコントロールブロックは，SYNフラグのTCPセグメントを受け取った時や，connectシステムコールがよばれた時に生成される．

MDOではこのようなイベントは生じないため，自動的にTCPコントロールブロックを生成する必要がある．CHから送信されてきたTCPセグメントのヘッダ部分を参照し，TCPコントロールブロックを動的に生成し，MNに成り変わって応答を送信する必要がある．

5.3 Proxyサーバ

MDOではHAがMNに成り代わって取得したTCPセグメントを，Proxyサーバによって一時的に保持する．MNがネットワークに再接続した際，Proxyサーバは保持していたTCPセグメントをMNに転送する．図7にProxyサーバのイメージを示す．

CHから送信されてきたTCPセグメントはmhsocketで受け取る．TCPセグメントが送信されてくる前のmhsocketは，HAのIPアドレス，9450番ポートという通常のソケットである．HAは，最初に送られてきたTCPセグメントのヘッダ部分を参照し，TCPコントロールブロックを生成し直す．そのため，Proxyサーバはmhsocketを利用して，TCPセグメントを取得できる．

CHから新たなTCPセグメントが送信されてきた場合，sock_entry構造体を新たに用意する．Proxyサーバは拡張したgetsockoptシステムコールを発行し，TCPセグメントのヘッダ部をもとにIPアドレス，ポート番号，シーケンス番号を保存する．また，TCPセグメントの管理を行っているプロセスのプロセスIDも保存する．sock_entry構造体はリンク構造になって保存される．図8にsock_entry構造体を示す．

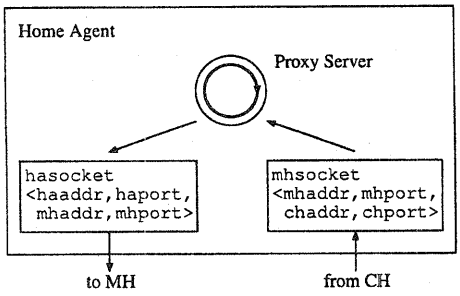

図 7: Proxyサーバの処理概要

```
struct sock_entry{
    LIST_ENTRY(sock_entry) sock_link;
    struct in_addr laddr;
                /* MN's IP address */
    u_short lport;
                /*      port num */
    struct in_addr faddr;
                /* CH's IP address */
    u_short fport;
                /*      port num */
    int pid;
            /* pid of listening from CH */
    tcp_seq iss;
    tcp_seq irs;
};
```

図 8: sock_entry構造体

Proxyサーバが保持しているTCPセグメントをMNに転送する場合，sock_entry構造体で保持されている情報を基に，MNへ転送するためのTCPコントロールブロックの内容を変更する．

しかし，Proxyサーバの資源も有限であるため，一時的にProxyサーバが保持できる上限を設定する必要がある．保持しているデータがある限度を過ぎた場合，ソケットから読み取ることを中断する．ソケットバッファが飽和状態になり，TCPはウィンドウサイズ0の応答を行う．そのため，CH

から送られてくるTCPセグメントをProxyサーバは中断させ，上限を設定することができる．

また，MNがネットワークに再接続した際，Proxyサーバは一時的に保持していたTCPセグメントを，MNに転送しなければならない．その際，Proxyサーバは`hasocket`を利用して送信する．

5.4 TCPオプションヘッダ

TCPは，ヘッダ部分のあて先IPアドレスとポート番号，および発信元IPアドレスとポート番号によって，どのソケットに渡すのかを決定している．

MNがネットワークに再接続した際，HAは`hasocket`を利用してTCPセグメントを転送する．しかし，通常のTCPヘッダの情報だけではMNは適切に処理できない．あて先IPアドレスとポート番号がHAのProxyサーバのものであり，CHのものとは異なる．このため適切なソケットにTCPセグメントに接続できず，通信が行えない．

MDOでは，ネットワークへの再接続時にHAが保持していたTCPセグメントをMNが受信し通信を再開できるように，CHのIPアドレスとポート番号をTCPオプションヘッダに記述する．MNはオプションヘッダからCHのIPアドレスとポート番号を得られ，転送されてきたTCPセグメントを適切なソケットに接続する．図9にMDOのTCPオプションヘッダを示す．

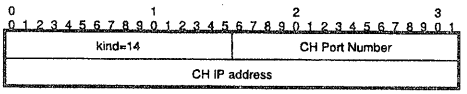

図 9: MDOのTCPオプションヘッダ

HAが一時的に保持しているTCPセグメントをMNに転送する際，HAはオプションヘッダを追加して転送する．オプションヘッダのついたTCPセグメントを受け取ったMNは，オプションヘッダに記述されているCHの送信元IPアドレスとポート番号により，適切なソケットに渡すことが可能になる．

MNはHAから転送されてきたTCPセグメントに対し，オプションヘッダを基にして，ヘッダの送信元IPアドレスとポート番号を変更する必要がある．変更されたTCPセグメントは，適切なTCPコントロールブロックを検索しソケットに渡せる．

また応答をHAに送信する際，あて先IPアドレスとポート番号をCHのものから，HAのProxyサーバのものに変更しなければならない．TCPでは通常，TCPコントロールブロックに記述されている，あて先IPアドレスとポート番号を利用して設定している．そのため，HAに応答を送信する場合は，ProxyサーバのIPアドレスとポート番号をヘッダに設定する必要がある．

5.5 タイマの停止

TCPでは，CHによるTCPタイマの切断だけではなく，MN自身のTCPタイムアウトによってコネクションが切断されてしまうことがある．

そのため，MNは自分がネットワークから切断されたと判断すると，TCPのタイマを停止し，タイムアウトによるコネクションの切断を防ぐ．また，MNがネットワークに接続したと判断すると，タイマを再開してTCPが保証する信頼性を回復する．図10にMNにおけるTCPタイマの状態遷移を示す．

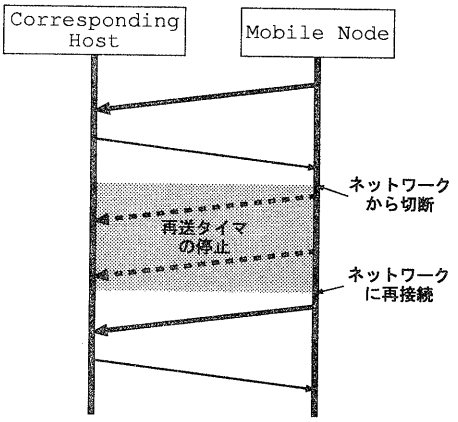

図 10: MNのTCPタイマの状態遷移

6 評価

6.1 測定環境

図11に示す環境でMDOの評価を行った．それぞれの計算機の性能を表1に示す．MDOのベースプロトコルであるMobile-IPは，Carnegie Mellon大学Monarchプロジェクト[9]の実装したMobile-IP ver1.1.0を用い，FreeBSD 2.2.2-RELEASEを使用して行った．また，それぞれの処理時間の測定には`microtime`を利用した．

6.2 定性的評価

関連研究であるI-TCP，Rover，Web Express，TCP-Rとの機能比較を行い，定性的な評価を行った．表2に性能評価を示す．

表 2: 関連システムとの定性的評価

	扱っている層	実装の必要な計算機	透過性 (IP 層)	透過性 (TCP 層)
I-TCP	TCP 層	MSRs(FA), MN	○	×
Rover	RPC 層	MN	×	×
Web Express	HTTP 層	MN	×	×
TCP-R	TCP 層	CH, MN	△	○
MDO	TCP/IP 層	HA, MN	○	○

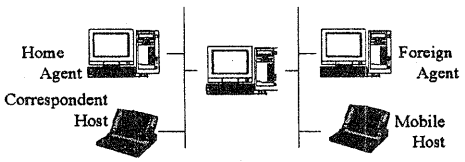

図 11: 測定環境

表 1: 各計算機の性能

計算機	CPU	ホスト名
HA	Pentium-Pro 200MHz	homed
FA	Pentium-Pro 200MHz	foreignd
MN	Pentium 130MHz	mobiled
CH	Pentium 120MHz	ch

TCP 層および IP 層における透過性を保証しているのは，MDO だけである．Rover や Web Express は，特定のアプリケーションを利用する場合にのみ透過性を保証しており，汎用性が低い．MDO は透過性を TCP/IP 層で保証しているため，アプリケーションを変更する必要がなく，汎用性が高いといえる．

IP 層における透過性を保証している I-TCP は，TCP 層における透過性を保証していない．そのため，移動型計算機が長時間ネットワークから切断されると，生成されていたコネクションが切断されてしまう．

TCP-R は，Mobile-IP を利用することで，MDO と同等に TCP/IP 層におけるの透過性を保証する．TCP-R は，三角ルーティングを回避する為，高い通信性能を実現できるが，CH での実装が必要である．MDO は，Mobile-IP を拡張することにより，HA，MH のみの実装で，TCP/IP 層における透過性を保証することが可能である．そのため MDO は TCP-R よりも実用性が高いといえる．

このような点から，MDO が関連研究に比べ TCP 層における透過性を実用性を保ちつつ，汎用的に保証している点で優れていることがわかる．

6.3 各機能の処理時間

MDO の TCP 層における処理に要する時間を測定した．測定結果を表 3に示す．

表 3: TCP 層における処理時間

処理内容	処理時間
HA のオプションヘッダ	0.08 ミリ秒
MN のオプションヘッダ	0.01 ミリ秒
TCP コントロールブロック	0.05 ミリ秒

HA での MDO のオプションヘッダを作成する処理に要する時間は，0.08 ミリ秒であった．

また，HA から転送されてきた TCP セグメントのオプションヘッダを，MN が処理するための時間は，0.01 ミリ秒であった．

Proxy サーバから MN に転送するすべての TCP セグメントに，HA は MDO のオプションヘッダを用意する．HA から転送されてきた TCP セグメントに対し，MN はすべて MDO のオプションヘッダの処理を行う．

しかし，この処理時間では TCP 層での処理に大きな影響を与えず，MDO による通信性能の低下はないことがわかった．

また，HA が MN に成り変って TCP セグメントを取得する際，TCP コントロールブロックを作成しなければならない．TCP コントロールブロックの作成は，CH からの TCP セグメントを初めて取得する際に行われる．この処理に要する時間は 0.05 ミリ秒であった．

これら処理時間の測定の結果，MDO の処理によって，TCP の通信に影響を与えず，通信性能が低下しないことがわかった．

6.4 再接続の処理時間

MDO では，MN がネットワークに再接続した際に，既に生成されているコネクションを継続して利用できる，そのため，MN がネットワークに再接続時に，どの程度で再接続が行われるかを測定した．測定結果を表 4に示す．

表 4: 再接続に要する時間

処理内容	処理時間
レジストレーションまで	0.01 秒
Proxy サーバからの転送まで	0.02 秒

MN 上のエージェントは，周期的にレジストレーションの処理を行うため，Mobile-IP のレジストレーションまでの処理には差が生じる．レジストレーションから，Proxy サーバが最初の TCP セグメントを転送してくるまでに要する時間は，ほぼ一定であり，0.01 秒であった．

この結果により，MN がネットワークに再接続した際に，円滑にコネクションを再利用できることがわかった．

7 今後の課題

現時点での MDO では，TCP によるタイムアウトを防いでコネクションを継続して利用できる．しかし，アプリケーションのタイムアウトに関しては考慮していない．そのため，アプリケーションのタイムアウトにより，コネクションが切断されてしまう場合がある．

そこで Proxy サーバにモジュール機能を追加し，HA が保持しているコネクションの処理に自由度を持たせる．CH から送信されてくる TCP セグメントのポート番号を基に，特定のアプリケーションにはモジュールを使用し，特別な処理を行えるようにする．

8 まとめ

移動型計算機の特徴である可搬性を活かすため，計算機が長時間ネットワークから切断して移動することを考慮したネットワーク環境が必要である．現在の移動型計算機のネットワーク環境に関する研究では，WWW など特定のアプリケーションの研究や，IP 層の透過性の保証や自動設定の研究が主であった．

しかし，これらの研究では，移動型計算機がネットワークの切断，接続に関係なく，継続して同じ環境を利用できない．移動型計算機がネットワークから切断されている間に，アプリケーションが強制終了する場合や再接続時にすでにコネクションが終了している場合がある．

本研究で設計，実装した MDO により，移動型計算機が長時間ネットワークから切断されても，それまで生成していたコネクションが TCP タイムアウトによって切断されず，再接続時に継続して利用することが可能になった．また，再接続する際の MDO による処理時間や再接続するまでに要する時間が小さい．そのため，TCP の性能をほとんど低下させずに MDO を利用できることがわかった。

これにより，長時間ネットワークから切断していた移動型計算機が再接続した際，アプリケーションを再起動してコネクションを再生成する必要がなくなる．また，移動型計算機による通信のために新たにアプリケーションを用意する必要もない．移動型計算機と通信を行う計算機の設定を変更する必要もない．つまり，MDO によって TCP 層における透過性が保証された．

参考文献

[1] Ajay Bakre etc. “Indirect TCP for Mobile Hosts”. 15th Int’l Conf. on Distributed Computing Systems (ICDCS), May 1995.

[2] Anthony D.Joseph etc. “Rover: A Toolkit for Mobile Information Access”. the 15th Symposium on Operating Systems Principles. December 1995.

[3] Henry Chang etc. “Web Browsing in a Wireless Environment: Disconnected and Asynchronous Operation in ARTour Web Express”. ACM MobiCom’97. September 1997.

[4] Daichi Funato etc. “TCP-R: TCP Mobility Support for Continuous Operation”.

[5] Charles Perkins. “IP mobility support”. Internet Request For Comments RFC 2002, October 1996.

[6] Charles Perkins. “IP Encapsulation within IP”. Internet Request For Comments RFC 2003, October 1996.

[7] Jim Solomon. “Applicability Statement for IP Mobility Support”. Internet Request For Comments RFC 2006, October 1996.

[8] James D. Solomon. “MOBILE IP The Internet Unplugged”. Prentice-Hall, Inc. 1998

[9] “CMU Monarch Project”, http://www.monarch.cs.cmu.edu/