2005－OS －98（11）
2005／2／23

ネットワークデバイス切替え時における通信の継続手法

奥田 勝己 †　瀧本 栄二 ††　毛利 公一 †††　大久保 英嗣 †††

† 立命館大学大学院理工学研究科
†† 株式会社 ATR 適応コミュニケーション研究所
††† 立命館大学情報理工学部

近年，有線や無線のネットワークデバイスを複数備えた小型で高性能な端末が急速に普及している．また，駅や空港，店舗などにおけるホットスポットの普及によって，ユーザが場所を問わず端末をネットワークに接続する環境が整いつつある．このような環境では，ユーザプリファレンスや利用可能なデバイスを適応的に切り替えながら通信を行うことが考えられる．しかし，IP を用いた通信では，デバイスの切替えやホストの移動後，通信を継続できないことが問題となる．これは，デバイスに割り当てられる IP アドレスが，接続するネットワークセグメントによって異なるため，デバイスを切り替えると IP アドレスが変更されることに起因する．本論文では，この問題を解決するために, 仮想デバイスを用いたルータへのトンネリング通信による通信手法について述べる，さらに, 仮想デバイスを用いた場合のオーバヘッドを補うために, プロセスの属性による経路選択機構についても述べる．

A Mechanism of Communication Continuation for Switching Network Devices

Katsumi OKUDA †　Eiji TAKIMOTO ††　Koichi MORI †††　Eiji OKUBO †††

†Graduate School of Science and Engineering, Ritsumeikan University
††Adaptive Communications Research Laboratories, ATR
†††College of Information Science and Engineering, Ritsumeikan University

Recently, mobile terminals that have multiple wired and wireless network devices are widely used, and users can access network everywhere. For example, host-spot with which users can connect his/her own terminal are available in stations, airports, stores, and so on. In such environments, the host may switch network devices, while processes are communicating with correspondents in remote hosts. The problem is that switching network device causes lost of existing communication session using IP. In this paper, we propose communication mechanism that uses virtual devices, and the route selection mechanism based on mobile attribute of user processes.

1　はじめに

近年，無線 LAN や Bluetooth が急速に普及し，携帯端末をネットワークに接続することが可能となった．また，有線や無線のネットワークデバイスを複数備えた端末も普及しつつある．このような環境では, ユーザプリファレンスや利用可能なデバイスを適応的に切り替えながら通信を行うことが考えられる．しかし, プロセスがコネクション指向のトランスポートプロトコルである TCP を用いて通信を行う場合, デバイス切替えの前後で通信コネクションを継続することができないという問題が発生する．これは，1 つのコネクションを, 通信するホスト双方の IP アドレスとポート番号の対として特定していることに起因する．通常，デバイスがネットワークのアドレスを保持しており，デバイスの切替は IP アドレスの変更を伴う．

この問題を解決するため，我々は，ネットワークデバイスと IP アドレスの対応をプロトコルスタックから隠蔽し，仮想デバイスに IP アドレスを与えることによる通信の継続手法の検討と実装を行っている．本論文で提案する手法では，カーネル内部のプロトコルスタックへの拡張を行わず，デバイスドライバによって仮想デバイスを実現することにより，通信の継続を実現する．提案手法では，仮想デバイスに対して永続的な IP アドレスを与え, これを利用して通信する．本手法では，仮想デバイスを，イ

ンターネットから到達可能なルーティング機能を持つホストとの間で VPN (Virtual Private Netowrk) 接続を行うことによって，永続的な IP アドレスの使用を実現する．また，本手法では，ユーザプロセスの分類による経路選択機構をオペレーティングシステム (以下, OS と記す) に組み込むことによって，仮想デバイスの利用によるによるコストを削減することを可能とする．

2 通信の継続手法

2.1 想定環境

本論文で想定するネットワーク環境を以下に挙げる．

(1) デバイス切替え後のホストは，インターネット上の他のホストに対して TCP のコネクション要求を行うことができる．
(2) デバイスの切替えやホストの移動は，アプリケーションから透過的に行う．
(3) デバイス切替えを行うホストの通信先では，本手法による機構の実装を必要としない．

(1) は，ホストがデバイス切替え後にファイアウォール内にある任意のネットワークに接続された場合においても，インターネット上の任意のホストに対して TCP のコネクション要求を発行できることを意味する．提案手法では，ホストにプライベート IP アドレスが割当てられている場合においても，ルータ上の NAT や IP マスカレードによって, ファイアウォール内部から TCP コネクションを確立することができることを前提とする．一般にオフィスやキャンパスに構築されるネットワーク環境では，この要求を満たす場合が多い．

(2) は，アプリケーションが，ホストのネットワークデバイスの切替えや移動を検知する必要がないことを意味している．通信の継続のための仕組みが提供されない場合，デバイスの切替え後も通信を継続したいアプリケーションは，デバイス切替えを検知し，コネクションを再確立するなどの対応が必要となる．あるいは，モビリティをサポートするライブラリなどを用いて通信する必要がある．提案手法では，TCP/IP による通信を行う既存のアプリケーションに対して変更を加えることなく適用可能とする．

(3) は，デバイスを切替えるホストと通信を行うホストは，通常の TCP/IP のプロトコルスタックのみを実装していればよいことを意味する．このため，一般のホストとも継続した通信が可能である．

2.2 永続的な IP アドレスの利用

通常, IP アドレスは，ホストのネットワークデバイスに対して割り当てられる．割り当てられる IP アドレスは，参加するネットワークセグメントのネットワークアドレスをプレフィックスとして持つアドレスである．したがって，あるホストがデバイスの切替えやネットワークセグメントの移動を行った場合，ネットワークデバイスに割り当てられる IP アドレスは，以前と異なる値となる．また，コネクション指向のトランスポートプロトコルである TCP は，通信する双方の IP アドレスとポート番号の対によって 1 つのコネクションを特定する．したがって， TCP を用いて通信を行っているプロセスが存在するホストが，デバイスの切替えやネットワークの移動を行った場合，コネクションを継続することが不可能となる．また，UDP などのコネクションレスなプロトコルを用いて通信を行っている場合であっても，アプリケーションレベルのセッションを継続することができない．

提案手法は, ホストが, デバイスの切替え後も継続して以前と同じ IP アドレスを持つことを実現することによる通信の継続手法である．このために, デバイス切替えを行うホスト上に, 仮想デバイスを設置し, 当該仮想デバイスに対して永続的な IP アドレスを与える．

2.3 システム構成

2.4 仮想デバイスの概要

本システムでは，デバイス切替えを行うホストとインターネットからアクセス可能なホストに以下に示す機構を実現する．また，これらが構成するネットワークを図 1 に示す．

- **ホスト仮想デバイス**
  移動やデバイス切替えを行うホストに存在する仮想デバイスであり，永続的な IP アドレスを保持する．
- **ホーム仮想デバイス**

IP パケットを転送可能なルータに設置する仮想デバイスである．ホーム仮想デバイスは，ルーティング機能をもったホストに設置される．

- **経路選択機構**
  特定プロセスに対して適切な経路の制御を行う．

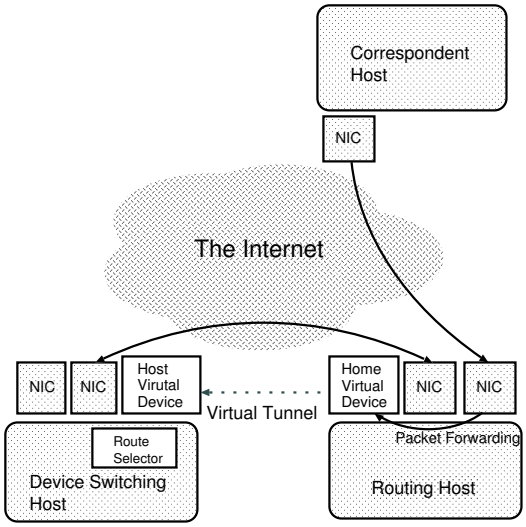

図 1　システム構成

本システムでは，ホスト仮想デバイスとホーム仮想デバイスの間に TCP を用いた VPN 接続を確立する．ホスト仮想デバイスとホーム仮想デバイスの通信に TCP を用いる理由は，2.1 節で述べたように，ホストがファイアウォールの内側に接続された後も，ホスト仮想デバイスからホーム仮想デバイスへのコネクション接続が可能な場合が多いためである．

本手法では，デバイスの切替えやホストの移動後も，ホスト仮想デバイスとホーム仮想デバイス間の VPN 接続を維持する．また，ホーム仮想デバイスは，ルーティング機能を持つホストに設置される．このため，ホスト仮想デバイスを有するホストは，VPN を経由したインターネットへの接続が可能である．すなわち，ホスト仮想デバイスを持つホストは，接続されているネットワークや使用しているデバイスの影響を受けることなく，常に仮想デバイスの IP アドレスを送信元とする IP パケットを送信することが可能である．また，仮想デバイスの IP アドレスを宛先とする IP パケットは，常にホーム仮想デバイスを持つルータに到着する．ホーム仮想デバイスは，ホスト仮想デバイスの IP アドレスを宛先とする IP パケットを TCP のデータストリームにカプセル化し，ホーム仮想デバイスに配送する．

3　仮想デバイスの構成手法

仮想デバイスは，ソフトウェアによって実現する擬似デバイスであり，通常のネットワークデバイスと同等のインタフェースをプロトコルスタックに提供する．仮想デバイスは，ネットワーク機構の一部であるデバイスドライバとして実現可能である．

デバイスドライバは，プロトコルスタックから渡されたパケットをデバイスに対応するイーサネットなどのデータリンクレイヤで定義されたフレームでカプセル化しデータを送信する．また，ネットワークデバイスが受信したフレームからデータをデマルチプレクスし，パケットが対応する上位のプロトコルスタックに対してデータを渡す．

実際のデバイスに対応するデバイスドライバは，パケットをカプセル化したフレームをデバイスの送信バッファ上にコピーし，デバイスに対してコマンドを発行することによってデータの送信を行う．また，デバイスにデータが到着した場合，デバイスからの割込みを契機としデバイスの受信バッファからデータをプロトコルスタックに渡し，対応するプロトコルのコードの実行を促す．これに対し，仮想デバイスに対応するデバイスドライバは，実際のデバイスを直接操作することはない．本手法では，仮想デバイスは，プロトコルスタックから渡されたパケットを TCP のデータストリームにカプセル化する．すなわち，アプリケーションから渡されたデータは，一度プロトコルスタックと仮想デバイスを経由した後，再度プロトコルスタックと実際のデバイスを経由することにより送信される (図 2)．

3.1　要求機能

ホスト仮想デバイスとホーム仮想デバイス間が構築する VPN は，ホスト仮想デバイスが永続的な IP アドレスを使用することを実現するための通信経路である．一般的な仮想デバイスを用いた VPN では，インターネットを介して外部から LAN に接続することを目的として構築される．このため，実際のデバイスが利用可能な場合のみ仮想デバイスも利用可能であればよい．一方，本手法で利用する仮想

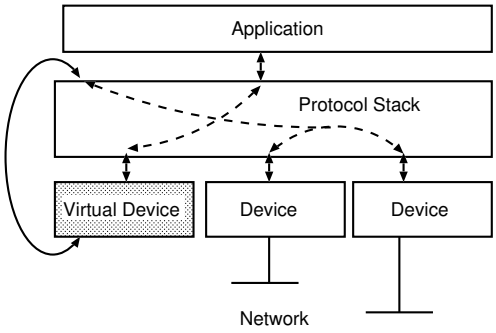

図 2　仮想デバイス

デバイスは，デバイスが永続的な IP アドレスを利用することを目的とする．したがって, ホストがデバイスの切替え時やホストの移動時によって，一時的にネットワークと切断された場合であっても，仮想デバイスは，見かけ上，プロトコルスタックから利用可能な状態でなければならない．すなわち，プロトコルスタックは，実際のデバイスがネットワークから切り放されている場合でも，仮想デバイスからデータを送信することが可能な状態である．これは，プロトコルスタックのコードが経路到達不能となるエラーを返すことを防ぐためである．

3.2　ホスト仮想デバイス

ホスト仮想デバイスは，デバイス切替えホストに存在する仮想デバイスであり，クライアントとして動作する．ホーム仮想デバイスとホスト仮想デバイスで送受信する IP パケットは，TCP のデータストリームでカプセル化して転送する．

ホスト仮想デバイスは，以下に挙げる契機でホーム仮想デバイスに対してコネクション要求を行う．

- ホーム仮想デバイスの初期化
- 実デバイスの切替えもしくはホストの移動時

OS は，通常のデバイスドライバの初期化と同様の操作によってホスト仮想デバイスを初期化する．このとき, ホスト仮想デバイスは，初期化時にホーム仮想デバイスとの間で IP パケットをカプセル化したストリームデータの通信路となるコネクションを確立する．

デバイスの切替えやホストの移動が発生した場合，ホスト仮想デバイスは，利用する IP アドレスが変わるため，既存の TCP コネクションを継続することはできない．このため，ホスト仮想デバイスは，デバイスの切替えを検知した後，切替え前に使用したコネクションのための資源を解放する．

3.3　ホーム仮想デバイス

ホーム仮想デバイスは，ルーティング機能を持ったホスト上に設置されるデバイスである．ホーム仮想デバイスは，初期化要求が到着すると，TCP の特定ポートでホスト仮想デバイスからの接続要求を待つ．

クライアントのデバイス切替や移動が起こった場合，ホーム仮想デバイスは，これを直接検出することは不可能である．このため，ホーム仮想デバイスは，クライアントからの再接続要求の受信によってこれをクライアントの実デバイスの切替えとして検出する．ホーム仮想デバイスは，ホスト仮想デバイスと同様に以前のコネクションの利用は不可能となるため，このコネクションの資源を解放する．

4　プロセスによる経路制御

本論文では，3 章までに，デバイスの切替えやホストの移動時における通信継続のための仮想デバイス間の VPN 接続による手法について述べた．仮想デバイスは, 通常のネットワークデバイスとしてのインタフェースを提供する．したがって, デバイス切替えを行うホスト上には, 実デバイスと仮想デバイスが存在する．本手法では, デバイス切替え後も通信を継続するためには, プロセスからプロトコルスタックに渡されたデータは, すべて仮想デバイスを介した VPN 経路を通る必要がある．

通常のネットワーク機構は，インターネットにデータを送信する場合，ルーティングテーブルのデフォルトゲートウェイを選択し，ゲートウェイとリンクレベルでの接続を持つ送信デバイスを決定する．このため，通常のネットワーク機構では, ホーム仮想デバイスに割当てられた IP アドレスをデフォルトゲートウェイとして設定する必要がある．これにより，ホストからインターネットに送信されるすべての IP パケットが VPN を経由する．

しかし，すべてのアプリケーションがデバイス切替えやホストの移動に対応する必要があるとは限らない．例として，ブラウザやメーラのように，短時間のうちにサーバとのセッションを終了するような

アプリケーションは，デバイス切替えやホストの移動後も通信を継続する必要はないといえる．また，同じアプリケーションでもセッションを確立している時間はユーザの意図によって異なる．telnet のようなプログラムでは，ユーザが短時間リモートホストにログインする場合もあれば長時間ログインしている場合もある．ユーザが長時間ログインするような場合であれば，デバイスの切替え後も通信を継続する必要があるが，短時間のうちにセッションを終了するのであれば，通信を継続する必要があるとは限らない．このため，すべてのプロセスが，通信の継続のために VPN 経路を介してデータの送受信を行う必要はないといえる．したがって，アプリケーションがモビリティを持つか否かだけでなくプロセスの実行時にモビリティが必要であるか否か選択可能である必要がある．

プロセス単位で通信データの経路を決定するためには，OS がこの機能を持つ必要がある．本手法では，OS が特定のプロセスに対する通信の継続を必要とするか否かの設定をするプリミティブを提供することによってアプリケーションの起動時に VPN 経路用いるか否かを決めることを可能とする．

4.1 プロセス単位のモビリティ

本手法では，OS は，プロセスごとに送受信する IP パケットをホスト仮想デバイスから VPN を経由して送信するか否かを決定するために，以下に示すデータを管理する．

- ホストの MOBILE 経路
- プロセスの MOBILE 属性

ホスト仮想デバイスが設置されたホストでは，IP パケットをホスト仮想デバイスを介して送受信することによって，デバイスの切替えやホストの移動時に通信を継続することができる．この VPN を介した経路を MOBILE 経路と定義する．

MOBILE 属性は，特定プロセスが，MOBILE 経路を用いるか否かを表す属性である．したがって，OS は，プロセスごとに MOBILE 属性を持つ．

4.2 ホストの MOBILE 経路の設定

ホスト仮想デバイスのドライバは，ホーム仮想デバイスとの間に VPN を確立する際，システムに MOBILE 経路の設定を行う．本手法では，OS は，MOBILE 経路の設定を行う以下のシステムコールを提供する．

```
int inet_setmroute(struct sockaddr_in *gateway);
```

inet_setmroute システムコールは，引数 gateway の IP アドレスを MOBILE 経路のゲートウェイに設定する．ユーザは，ホスト仮想デバイスの起動後，**inet_setmobile** を発行することによってシステムに MOBILE 経路の設定を行う．

4.3 プロセスの MOBILE 属性の設定

OS 内部のプロセステーブルエントリは，MOBILE 属性をパラメータとして持つ．MOBILE 属性が ON の場合，当該プロセスは，デバイス切替え後も通信の継続が必要であることを意味する．プロセスが **fork** の発行によってプロセスを生成した場合，プロセステーブルエントリの MOBILE 属性は，子プロセスに引き継がれる．

MOBILE 属性は，以下に示すシステムコールによって設定する．

```
int inet_setmobile(int pid, int flag);
```

inet_setmobile システムコールは，引数 **pid** をプロセス ID として持つプロセステーブルエントリの MOBILE 属性を引数 **flag** の値にセットする．

4.4 経路選択機構

in_setmroute は，引数で与えられた IP アドレスから OS 内部にルーティングテーブルエントリと同型のデータを作成する．作成されるデータの構造を表 1 に示す．

表 1　inet_setmroute が生成する MOBILE 経路

宛先	ゲートウェイ	デバイス
0.0.0.0	ホーム仮想デバイス IP アドレス	ホスト仮想デバイス

BSD の TCP/IP 実装では，プロセスが **connect** システムコール呼出しによって TCP のコネクション要求を発行する際，当該コネクションのホスト側 IP アドレスを決定する．そして，ホスト側の IP アドレスを決定するため，通信先 IP アドレスをキーとしてルーティングテーブルが探索される．TCP のコードは，このとき発見したルーティングテーブルエントリをこのコネクションに関連付ける．

本手法では，**connect** システムコールの呼出しプロセスが，MOBILE 属性を持っている場合，ルーティングテーブルから探索した経路を破棄し，これを **inet_setmroute** によって設定された MOBILE 経路で置き換える．したがって，コネクションのローカル側の IP アドレスは，ホスト仮想デバイスの IP アドレスとなる．また，当該コネクションに関連する送信パケットは，すべて VPN 経路を経由する．

4.5 利用例

本手法では, プロセスごとに MOBILE 属性を設定し, MOBILE 属性が必要なプロセスだけが, デバイス切替え時の通信継続に対応することができる. 本節では, 利用例を用いてこれを示す. 以下, 例として **telnet** コマンドの起動時におけるモビリティのサポートについて述べる. ユーザは，**telnet** が通信の継続を必要としない場合，コマンドラインから

```
% telnet remotehost
```

と入力することによってプログラムを実行する．一方，通信の継続のため VPN 経路を用いる場合，

```
% mobile telnet remotehost
```

と入力することによってプログラムを実行する．

mobile コマンドは, 自身のプロセスに MOBILE 属性をセットし, 引数で与えられたコマンドを実行するプログラムである．mobile コマンドの簡単な実装を図 3 に示す.

```
int main(int argc, char *argv[])
{
    int i;

    /* モバイル属性の設定 */
    inet_setmobile(getpid(), 1);

    for (i = 0; i < argc - 1; i++)
        argv[i] = argv[i + 1];
    argv[i] = NULL;

    /* プログラムの実行 */
    execvp(argv[0], &argv[0]);

    return 0;
}
```

図 3　mobile コマンド記述例

5 実装と評価

5.1 ホスト仮想デバイス

現在, 我々は, 適応的資源管理を行う OS である AG[1] の開発を行っている．今回, ホスト仮想デバイスの実装を AG 上で行った．AG のネットワーク機構である AGNET は，4.4 BSD のプロトコルスタックをベースとするライブラリであり，AG カーネルとリンク可能である．AGNET は，専用の仮想デバイスを用いることによって，FreeBSD もしくは Linux のアプリケーションのネットワークライブラリとしても利用可能である．この場合，プロトコルスタックをユーザレベルで動作させることが可能である．また，AGNET は，AG カーネル内部においてもシステムスレッドによる socket の利用を可能としている．

今回の実装では，仮想デバイスは，仮想イーサネットドライバとして実現している．仮想デバイスは，プロトコルスタックから呼び出されるコードと，2 つの送信スレッドと受信スレッドから構成される．送信スレッドは，デバイス構造体のメンバにキューイングされたパケットをイーサフレームでカプセル化し，TCP でこれをホスト仮想デバイスに送信する．受信スレッドは，ホスト仮想デバイスから送信されるデータを read する．データがない場合，read はブロックする．パケットが到着しブロックが解除されると，受け取ったデータを対応するプロトコルの入力キューにキューイングし，プロトコルスタックのスレッドを起こす．プロトコルスタックは，パケットを送信する場合，仮想デバイスの入力キューにパケットをキューイングし，仮想デバイスの送信スレッドを起こす．送信スレッドと受信スレッドは, 同一のコネクションの資源を扱うため, 排他的に動作する.

5.1.1 ホーム仮想デバイス

ホーム仮想イーサネットデバイスは，FreeBSD もしくは Linux で実行可能なユーザプログラムであり，tap ドライバ [2] を用いて実装している．tap デバイスは，2 つのインタフェースを提供している．tap デバイスは，疑似デバイスであり，通常のデバイスと同様にイーサネットのネットワークインタフェース

としてプロトコルスタックから扱われる．また，ユーザプロセスは，デバイスファイル /dev/tap に対して読出しと書込みができる．/dev/tap に対して書き込まれるデータは，tap デバイスに到着したイーサフレームとして扱われ，カーネルのプロトコルスタックに渡される．また，カーネルのプロトコルスタックが tap デバイスに渡したデータは，tap ドライバでイーサフレームにカプセル化され, /dev/tap から読み出しが可能である．

ホーム仮想デバイスは，ホスト仮想デバイスから受信したイーサフレームをそのまま, /dev/tap に書き込むことによって，カーネルのプロトコルスタックにデータを渡す．また, /dev/tap を読み出し, プロトコルスタックから渡されたパケットをイーサフレームにカプセル化し, ホーム仮想デバイスに送信する．

5.2 評価と考察

ホスト仮想デバイス間とホーム仮想デバイス間の経路における IP パケットの TCP ストリームによるカプセル化におけるデータの増加量を計測した．ホスト仮想デバイスが存在するホスト AG からホーム仮想デバイスを経由する経路を用いて通常の FreeBSD のホストにデータを送信する．また，実デバイスの送受信バイト数を計測するため，ホーム仮想デバイスが存在するホストとホーム仮想デバイスが存在するホストは，イーサネットケーブルで直接接続している．計測に利用したアプリケーションは，discard サーバに接続し，ソケットに対し合計 1 M バイトのデータを 1024 バイトずつ書き込む．表 2 は，それぞれプロセス，仮想デバイス，実デバイスが送受信したデータ量である．

表 2　入出力バイト数

	送信バイト数	受信バイト数	送受信バイト数
プロセス	1048576	0	1048576
仮想デバイス	1086492	25592	1112084
実デバイス	1203486	113828	1317314

仮想デバイスと実デバイスの入出力バイト数の差が仮想デバイス間の VPN 経路による増加分である．プロセスが受信していないのに対して, 仮想デバイスおよび実デバイスが受信を行っているのは，TCP の ACK セグメントや VPN 維持のためコントロール用に用いたデータを受信しているからである．仮想デバイスが合計 111,2084 バイトを送受信したことに対し，実デバイスは，合計 131,7314 バイトを送受信し，約 18 % 増加する．

本手法では，4 章で述べた経路選択機構を用いることにより，余計なデータ転送を伴う通信を MOBILE 属性を持ったプロセスだけに限定することが可能である．

今回の実装では，ホスト仮想デバイスとホーム仮想デバイス間の VPN で転送するデータに対して，圧縮を行っていない．VPN における IP パケットを適切なアルゴリズムで圧縮することによって，VPN 間のトラフィックを軽減することが可能である．この場合，ホスト仮想デバイスからの IP パケットの送信に，TCP のコードによるオーバヘッドに圧縮するための処理が加わるため, 圧縮を行った場合の実装を行い検討を行う必要がある．また，現在，仮想デバイスの起動後，IP アドレスを手動で割り当てているが， DHCP による割当ても可能である．

6 関連研究

移動透過性を実現するための IP への拡張として，RFC 2002 で定義されている Mobile IP があげられる [3]．Mobile IP では，移動ノード，ホームエージェント，外部エージェントを用いる．移動ノードは，常にホームアドレスを送信元アドレスとしてパケットの送信を行う．移動ノードは，移動ノードがホームリンクを離れて外部リンクに存在する場合，移動ノードのホームアドレス宛のパケットは，ホームエージェントが移動ノードに転送を行う．Mobile IP では，エージェントの探索やエージェントへの移動ノードの登録に ICMP への拡張，ホームエージェントから外部エージェントへのトンネリング通信のための，IPinIP カプセル化など，カーネル内のプロトコルスタックに変更を必要とする．一方，本手法では，既存の TCP/IP プロトコルスタックに拡張を加えることなく，仮想デバイス間の VPN によって同様の移動透過性を実現する．仮想デバイスは，ホスト OS 上のデバイスドライバとして実装されるため，TCP/IP プロトコルスタックを持つ OS であれば容易に対応することができる．また，Mobile IP

では，移動ノードがホームリンクを離れた場合に必要とする気付けアドレスは，ホームエージェントから到達可能でなければならない．外部リンクのルータ上に外部エージェントが存在する場合，移動ノードが利用する外部エージェント気付けアドレスは，ホームエージェントから到達可能である．しかし，外部エージェントが存在しない場合，移動ノードが利用する共存気付けアドレスは，グローバルアドレスである必要がある．したがって，移動ノードがプライベートアドレスを利用しなければならないネットワークセグメントに移動した場合，通信を継続することができない．本手法では，デバイスに割当てられたアドレスがプライベートアドレスを利用する場合であっても，IP マスカレードによってインターネットに TCP のコネクション要求を行うことが可能であれば通信を継続することができる．

MSOCKS[4] は，ライブラリによる，TCP レベルの通信の継続を実現している．MSOCKS は，通信する 2 つのホストの間にプロキシを設置する．MSOCK ライブラリは，通信先へのコネクション要求をプロキシへのコネクション要求に置き換えることにより，プロキシとの間にコネクションを確立する．同時にプロキシは，通信先との間に TCP コネクションを確立する．デバイスが切り替わった場合, MSOCKS ライブラリは, プロキシとのコネクションを再確立する．プロキシは, 以前のコネクションとのマッピングを通信先から透過的に行うことにより, 通信先とのコネクションを継続させる．

Mobile Socket[5] は，通信の継続を実現する Java のクラスライブラリである．ホストの移動よって IP アドレスが変わった場合，TCP のコネクションを再確立しこれをライブラリで隠蔽する．Mobile Socket では，ホストの移動後，相手ホストに対してコネクション要求を出さなければならないため，通信する双方のホストが移動する場合，通信を継続できない場合が存在する．

7 おわりに

本論文では，デバイス切替え時における通信の継続手法について述べた．本手法では，仮想デバイス間で構築する VPN を通信経路として IP パケットを送信することにより，通信の継続を可能とする．また，通信を行うプロセスに MOBILE 属性を持たせることによって，プロセス単位でのモビリティの実現を可能とする．これにより，VPN を通信経路とする IP パケットの送受信をデバイス切替に必要なプロセスのみに限定することが可能である．

現在の実装では，仮想デバイスをイーサネットデバイスとしているため，TCP のデータストリームでカプセル化されるパケットは，イーサフレームのヘッダを含むオーバヘッドが生じている．また，カプセル化するデータに圧縮を行っていない．このため，仮想デバイス間のデータ転送量は，縮小できる可能性がある．

参考文献

[1] 瀧本栄二, 芝公仁, 大久保英嗣: “エージェント技術を用いた分散オペレーティングシステムの構成手法”, 情報処理学会システムソフトウェアとオペレーティングシステム研究会 2002-OS-89, Vol. 2002, No. 13, pp. 117-123 (2002).

[2] http://vtun.sourceforge.net/tun/

[3] ジェイムス・D・ソロモン著, 寺岡文男・井上淳監訳: “詳説 MobileIP 移動ノードからのインターネットアクセス”, プレンティスホール (1998).

[4] Maltz, D. and Bhagwat, P.: “MSOCKS:An Architecture for Transport Layer Mobility”, Proceedings of the Seventeenth Annual Joint Conference of the EEEE Computer and Communications Societies, pp. 1037–1045(1998).

[5] Tadashi Okoshi, Masahiro Mochizuki, Yoshito Tobe, and Hideyuki Tokuda: “MobileSocket: Toward continuous operation for Java applications”, In Proc.IEEE International Conference on Computer Communications and Networks, pp. 50–57(1999).