SCTPのIPアドレス自動更新機能におけるアドレス取捨選択機構

丸山 伸 †, 小塚 真啓 ††, 中村 素典 ‡, 岡部 寿男 †

† 京都大学学術情報メディアセンター †† 京都大学大学院法学研究科

‡ 大学共同利用機関法人情報・システム研究機構国立情報学研究所

近年、複数の通信用インタフェイスを備える端末が増えていることや、IP version 6 の普及が進んでいることなどの影響により、端末に同時に複数のアドレスが割当てられるようになっている。この複数のアドレスを切り替えつつ利用することにより、通信経路を冗長化でき信頼度を高めることが出来る。ところが、端末が取得したアドレスの全てを利用しようとすると、そこにはポリシー上の問題により利用するべきではないアドレスやプライベートアドレス等の相手と通信できないアドレスが含まれるため、無駄なタイムアウトが生じたり通信を維持できなくなったりする可能性がある。次世代のトランスポートプロトコル SCTP は通信の両端に同時に複数のアドレスを利用でき、通信を開始した後においても端末の持つアドレスに生じた変化を既存の接続にも反映させる機構 (ADDIP 拡張) を持つことから複数のアドレスを利用した通信経路の冗長化に適しているが、SCTP にも同様の問題が存在した。
そこで本研究では、アドレスを取捨選択するポリシーを事前に与えておき、アドレスの取得時にこのポリシーに照らして通信に利用しようとするべきかどうかを判断できるようにする手法を提案する。Protocol Independent なプログラムスタイルで記述されていることにより SCTP を利用するプログラムが端末に割当てられた全てのアドレスを利用するように記述されていた場合にも、利用するべきではないアドレスをこのポリシーに照らして事前に除去出来る。また、SCTP の ADDIP 機構を拡張し複数のアドレス変更情報をまとめて通知できるようにすることで、相手と通信できない不適切なアドレスをポリシーにより除去できなかった際にも、通信を維持できるようにした。

A study of the filtering and picking mechanism of IP addresses over the automatic update function of IP addresses in SCTP

Shin Maruyama † Masahiro Kozuka †† Motonori Nakamura ‡ Yasuo Okabe †

† Academic Center for Computing and Media Studies, Kyoto University

†† Graduate School of Law, Kyoto University

‡ National Institute of Informatics

Recently, an endhost may have multiple interfaces for communication and IP version 6 has come to the stage for deployment, an endhost is often to have multiple IP addresses at the same time. Using these addresses for the purpose of redundancy, the reliability of the communication is enhanced. But using all the addresses which the endhost acquire implies a problem of using addresses which do not suit the policy or which are private addresses, and results in unnecessary timeout or disconnection of the communication. The next generation transport portocol "SCTP (Stream Control Transmission Protocol)" is capable of handling multiple address on both endhost, and also capable of making modification of IP addresses on both ends of existing connection by using ADD-IP extension. With these features, SCTP is very suitable for making the path of communication redundant and stable by using multiple addresses, but SCTP also implies the same problem.
On this research, we propose a new feature of matching the acquired address to the "address usage policy" prior to using the address, and make the host can judge which address should be used or ignored. In case that the SCTP application is written with Protocol Independent Programming style, the application will use all of the reported address changes caused on the endhost, improper addresses are removed beforehand.
Another important improvement on ADD-IP mechanism to send multiple Address Change Configuration chunks (ASCONF) at a time makes the communication robust and stable against the case that an improper address happens to pass the filter of the "address usage policy".

1 はじめに

近年、複数の通信用インタフェイスを備える端末が増えていることや、IP version 6 の普及が進んでいることなどの影響により、通信を行う端末に同時に複数のネットワークが接続され、同時に複数アドレスが割当てられることが増えている。端末が複数のネットワークに同時に接続している時には、これらを冗長な経路として利用することで、通信経路の障害に対する耐性が増し通信の信頼度を高めることが出来る。その一方、これらのネットワークには有線か無線かによる特性や帯域の違い、通信量による課金の有無、運用ポリシーによる制約などの特徴があるため、必ずしも端末に割当てられた全てのネットワークやアドレスを利用して良いとは限らず、ユーザ毎やアプリケーション毎にこれらを取捨選択する必要が生じている。

ところで、近年注目されている SCTP "Stream Control Transmission Protocol (RFC2960)"[1] は通信を行う両端の端末において同時に複数のアドレスを利用できることを特徴とするトランスポートプロトコルであり、複数のアドレスを同時に利用した通信に適している。

SCTP は、通信の開始時に利用するアドレスを相互に交換するだけでなく、通信を開始した後におけるアドレスの増減を既存の接続にも反映させる機構 (ADDIP 拡張[2]) を持つが、既存の ADD-IP 拡張とそれを利用するための SocketAPI 拡張では端末の持つアドレスを選択的に利用する機構が不十分である。一般に ADD-IP 拡張を利用するプログラミングでは AUTO-ASCONF と呼ばれるフラグを立てて、端末のアドレス更新を既存の接続にも自動的に反映させようとする手法が利用されるが、この手法ではアドレスを選択的に利用することは出来ない。このフラグを立てずに端末が取得したアドレスを利用するかどうかをプログラムにより判断する手法も考えられるが、そのためにはプログラムを大幅に拡張するか、ポリシーの変更毎に再コンパイルするといった作業が必要となり、このような SCTP の利用法は現実的ではない。

そこで、本研究においては SCTP におけるアドレス自動更新機構を拡張し、事前に設定されたポリシーに従ってアドレスを利用するか否かを判断する機構を提案する。まずアプリケーションの起動にあたっては環境変数により事前にポリシーを指定し、SCTP を利用する端末が新しいアドレスを取得した際にはそのアドレスをポリシーに照らし、アドレスを利用するかどうか判断する。ポリシーに適合した場合のみアプリケーションに対してアドレスが増えたことを通知するようにする。このようなアドレス自動更新機構を利用することで、あらかじめ指定されたアドレスのみを用いて通信を継続できるようになり、安定して通信を継続出来るようになることを示す。

また SCTP の ADD-IP 拡張に一旦全てのアドレスが無くなった後、新たに得たアドレスが相手と通信できないアドレスであった場合に、通信の再開までの時間が無駄に長くなったり接続 (アソシエーション) が切断されたりするという問題が存在した。そこで ADD-IP 拡張に修正を加えることにより、もし不適切なアドレスをアソシエーションに追加しようとした際にも、通信を維持できるようにした。

以下 2 章では、本研究で利用する SCTP およびその ADD-IP 拡張について述べ、3 章で SCTP で複数アドレスを取り扱う上での既存の API に存在する問題点について述べる。4 章では事前にポリシーを指定できるアドレス自動更新機構と、ADD-IP 拡張に存在する問題を回避するための改善手法について提案する。5 章ではこの提案に基づく実装を行い、意図通りに動作することを確認し本提案の有効性を示す。6 章では他のプロトコルとの比較を含めて結論を述べる。

2 SCTP と ADD-IP 拡張による複数アドレスの利用

SCTP "Stream Control Transmission Protocol (RFC2960)[1]" は複数のアドレスを利用できるトランスポートプロトコルとして標準化され、通信するエンドホストの両端において同時に複数のアドレスを利用できることを特徴としている。SCTP は接続（アソシエーション）の開始時にそれぞれのエンドホストが持つ IP アドレスの情報を交換し相互に保持する。

アソシエーションの開始時に相手に通知するアドレスはアプリケーション側から特に指定されない限り、エンドホストの持つ全てのアドレスとなる。アソシエーションの開始後に、インタフェイスの増減などによって IP アドレスの増減や変化が発生することもあるが、このような IP アドレスの増減や変化に対して RFC2960 に規定される SCTP は特に何も処理を行わない。

2.1 ADDIP 拡張による動的アドレス更新

SCTP はプロトコルの拡張を行いやすいという特徴がある。ADD-IP 拡張[2] はこの特徴を活かした RFC2960 に対する重要な拡張であり、この拡張を利用すると開始後のアソシエーションに対してアドレスの変更を行うことが出来る。

エンドホストの持つアドレスが増減した際には ASCONF(Address Configuration) と呼ばれるチャンク (データ) を送出する。ASCONF チャンクを受信したピアは、そのチャンクの受信とアドレス変更が適切に完了したこと

を通知するために、ASCONF_ACK チャンクを返送する。ASCONF を送信したエンドホストは、ASCONF_ACK チャンクを受信することで、変更されたアドレスを利用して通信を行うことができるようになる。

また、アドレス変更情報の順序を保ち両エンドホストが保持する IP アドレス情報の同期を取るために、ASCONF チャンクには Sequence Number と呼ばれる情報が含まれており、各エンドホストが送出する全ての ASCONF チャンクはシリアライズされる。各 ASCONF チャンクは Sequence Number の順に従って送出され、その番号の順に処理される。送出した ASCONF に対応する ASCONF_ACK を受信するまでは一定のタイムアウト毎に同じ ASCONF チャンクを再送し続け、次の ASCONF を送出してはならない規則となっている。

なお、ASCONF チャンクは第三者によるなりすまし攻撃（Man-in-the-Middle Attack）を防ぐため、ASCONF チャンクの正当性を保証する AUTH チャンク（このチャンクには HMAC が含まれる）[3] とともに送られなければならない。

2.2 SCTP と ADD-IP 拡張の利用

SCTP は、TCP や UDP を利用するために広く用いられてきた BSD Socket インターフェースに対応しており、TCP と互換性のあるプログラミングスタイルで SCTP を利用することができる [4]。Protocol Independent なプログラミングスタイル [5] で作成された TCP のアプリケーションでは、オペレーティングシステムが SCTP に対応するだけで、アプリケーションに一切の修正を行うことなく、SCTP を利用することも可能である。

ADD-IP 拡張に関しては、エンドホストで発生したアドレスの増減・変更に対応して ASCONF チャンクを自動的に送出する AUTO-ASCONF 機構が用意されている。AUTO-ASCONF の有効・無効は、ソケット毎に setsockopt() を用いて制御することができる。但し、AUTO-ASCONF を利用するためには、ソケットを INADDR_ANY アドレス（0.0.0.0）で割り当てなければならず、特定のアドレスのみを割り当てた場合には、エンドホストにおいてアドレスの増減・変更が発生したとしても、対応する ASCONF チャンクの送出は行われない。

3 複数アドレス利用に伴う問題

TCP を利用する既存のアプリケーションは、Protocol Independent プログラミングスタイルに従って作成されている場合には、オペレーティングシステムの変更を行うだけで、SCTP を利用することが可能である。

しかし、複数アドレスの割り当てや、AUTO-ASCONF を利用して、エンドノードで発生したアドレスの変化に対する ASCONF チャンクを自動的に送出するためには、INADDR_ANY アドレス（0.0.0.0）で割り当てを行わなければならない。このため、SCTP で利用するアドレスの取捨選択を行うことができず、また、エンドノードにプライベートアドレスやフィルタされたアドレスといった通信に利用できないアドレスが付与されている、あるいは、追加される場合には、そのような利用できないアドレスもアソシエーションに含まれてしまい、本来は不要なタイムアウト発生してしまう可能性がある。

AUTO-ASCONF を用いることなく、アプリケーションが割り当てられるアドレスを明示的に変更できるように、sctp_bindx() という API も用意されている。この sctp_bindx() を用いて、アプリケーションがアドレスの取捨選択を行うことも可能であるが、アプリケーションが、エンドノードに発生したアドレスの変更情報を取得し、その変更を反映させるようにするためには、大幅な改変が必要となる。また、アドレスの変更情報を取得する仕組みは、オペレーティングシステムごとに異なっており、オペレーティングシステムごとにプログラムを改変するという手間が生じかねない。

ADD-IP 拡張それ自体にも問題があった。ADD-IP 拡張の当初の設計では、新たに取得するアドレスが相手と通信可能なものであることが前提とされていた。移動などによって有効なアドレスが全て消えた後、一時的に、通信を行うことができないアドレスが追加された際には、ASCONF チャンクがその通信できないを Source アドレスとして送出されることになる。この場合、ASCONF を受信したピアは、ASCONF-ACK を通信できないアドレスに送り返すため、ASCONF-ACK の受信を行うことができず、それ以降いつまでも通信を再開できないという現象が生じていた。

4 提案手法

前章では、SCTP や ADD-IP 拡張を利用して複数アドレスによる通信を行う場合、既存の仕組みだけでは、エンドノードに属する全てのアドレスをアソシエーションに含めるか、あるいは、既存のプログラムの大幅な改変を行い、特定のアドレスを割り当てることになることを示した。また、AUTO-ASCONF を用いてプログラムの改変を行わず、エンドノードに属するアドレス全てを利用する場合、通信出来ないアドレスに対してアドレス変更情報を送信しようとすることにより、接続の切断が生じうることを示した。

本研究においては、SCTP を利用するアプリケーションを起動する際に、アドレスの取捨選択に関するポリシーをアプリケーション毎に事前に与えることで、アプリケーション起動後に得たアドレスを利用するかどうか

取捨選択し、その機構においてプライベートアドレスや利用するべきではないアドレスなどを排除することができるようにする手法を提案する。また、ポリシーにおいては利用可能とされたアドレスであっても、障害等が原因で実際には利用できないことがあるため、そのような状況においてもアソシエーションが切断されることがないよう、複数のASCONFチャンクを単一のパケットで送出する手法を提案する。

4.1 アドレス取捨選択ポリシー

アドレスの取捨選択ポリシーを構成する単位としては、エンドノード共通・ユーザーごと・プロセスごと・ソケットごとなど、色々な選択が考えられるが、本研究では、プロセスごとにポリシーを構成する手法を採用する。

プライベートアドレスなどの利用できないアドレスの排除は、全てのアプリケーションで共通であると考えられるが、利用する上で適切なアドレスは、アプリケーションごとに異なる。このような、アプリケーション単位で異なるポリシーを適用するためには、プロセス又はソケットごとにポリシーを構成すべきである。複数のSCTPアソシエーションを利用するようなアプリケーションの場合、ソケット単位で構成されたポリシーが適切である場合もあると考えられるが、個々のソケットをポリシーを定義する段階において識別することは困難である。

ポリシーは、マッチしたアドレスを捨てるリストと、マッチしたアドレスを拾うリストの2つから構成され、前者を適用した後、後者を適用して、最終的に取捨選択が決定される。このような構成を採ることで、プライベートアドレスを原則として排除しつつ、特定のプライベートアドレスを拾うことが可能となる。

ソケットへのアドレス割り当て時、及びアドレスの取得時に、エンドノードが有するアドレスがポリシーと照合され、アソシエーションに追加されるかどうか判断される。また、アドレスの削除においても同様の処理を行う。

このようなポリシー機構を導入することで、ポリシーに反するアドレスや明らかに通信を行えないアドレスに関するASCONFチャンクを送ることが防止できる。ポリシーに反するアドレスを利用してしまうことはなくなり、また通信を行えないアドレスを利用しようとすることによる無用なタイムアウト待ちが生じることも防ぐことが出来る。

4.2 Multiple ASCONF

アドレス取捨選択ポリシーを導入することで多くの場合において不要なアドレスに関するASCONFの送出を避けることが出来るが、通信を出来ない全てのアドレスを除去することにはならない。端末が全てのアドレスを一旦失ったあとで新たに得たアドレスがネットワークの障害等が原因で通信出来ないアドレスであった場合、そのアドレスに関するASCONF-ACKを受信することが出来ず、いつまでもASCONFの処理が完了しない。

一方、ADD-IPのプロトコルでは1つのASCONFの処理を完了するまで、それ以外のASCONFを送出してはいけないことになっているため、その次に生じたアドレス変更情報を相手に通知できないという問題が生じていた。

そこで1つのASCONFの処理が未完了の状態であった場合にでも次のASCONFを送出しても良いことにした上で、そのような際には未完了のASCONFも同時にまとめて（バンドルして）再送するようにプロトコルを変更する。ASCONFを受信した側はその複数のASCONFを順次処理する。その際に処理する一連のASCONFの中には、相手と通信可能なアドレスを追加するためのASCONFをも含むため、これまでのADD-IPの場合と異なり、このアドレスを利用してASCONF-ACKもまとめて送出することが出来るようになった。

また、個別のASCONFを処理する手順は従来のADD-IP拡張における手順からなんら変更されていないため、既存のプロトコルの特徴に影響を与えるものではない。

5 実装

本研究の効果を確認するため、FreeBSD 6.2用のSCTPカーネル実装[6]をベースとして、アドレス選択ポリシー機構の実装を行った。アドレス選択ポリシーは、プロセスごとにポリシーを設定することができるように、ソケットごとに作成されるsctp_inpcb構造体にポリシーを格納できる領域を持つ。SCTPカーネル実装は、複数アドレス及びアドレスの変化を以下のように取り扱う。sctp_inpcb構造体には、ピアに対して通知されるアドレスの一覧が含まれる。SCTPソケットの作成とINANY_ADDRアドレスを用いたソケットのバインドが行われると、sctp_inpcb構造体上のアドレス一覧に、ホストに付与されている全てのアドレスが追加される。異なるネットワーク環境への移動などによって、ホストに付与されたアドレスの追加・削除が発生すると、SCTPカーネル実装はルーティングソケット経由で、ホスト全体のアドレス一覧の変化を検出し、sctp_inpcb構造体上のアドレス一覧を更新する。この時、AUTO_ASCONF機構が有効となっていると、ア

ソシエーションそれぞれにおいて、対応したASCONFチャンクの送信が行われる。アドレス選択ポリシー機構は、ソケットのバインド及びホストに付与されたアドレスの追加・削除が発生した場合に、次のように動作する。INANY_ADDRアドレスを用いてソケットのバインドが行われると、アドレス選択ポリシー機構は、ホストが有するアドレスのそれぞれについて、sctp_inpcb構造体上のポリシーと一致するかどうかのチェックを行い、ポリシーに一致したアドレスだけをsctp_inpcb構造体のアドレス一覧に追加する。ホスト全体のアドレス一覧に変化が生じた場合においても、sctp_inpcb構造体上のポリシーと一致するアドレスの追加・削除だけが、sctp_inpcb構造体のアドレス一覧に反映され、ASCONFチャンクが送信される。

sctp_inpcb構造体上に作れるアドレス選択ポリシーを設定する方法として、プロセスごとに管理される環境変数を利用した。アドレス選択ポリシー機構はSCTPソケットの作成時にソケットを作成したプロセスの環境変数を参照し、"SCTP_ALLOW_ADDRESSES"と"SCTP_DENY_ADDRESSES"とを、sctp_inpcb構造体上のポリシー格納領域にコピーする。これら2つの環境変数には、IPアドレスが","で区切って列挙されており、一致した場合には、それぞれ許可と排除を行う。また、192.168.0.0/24のように、サブネット単位で記述することもできる。sctp_inpcb構造体上のポリシーは、one-to-oneのリスニングソケットから4-way handshakeを経て新たなソケットが作成される場合には、新しいソケット上に含まれるsctp_inpcb構造体上のポリシーへコピーされる。

これらの実験の結果、意図通りに動作することを確認した。

6 考察

端末に割り当てられた複数のアドレスを利用し、マルチホーミング・モビリティ通信を実現する手法としては、SCTP以外にも複数のCare of Address (CoA)を同時に利用するMobile IP拡張[7, 8]やHIP[9, 10]、shim6[11]が提案されている。

これらの第3層（IPレイヤー）を拡張する手法においては、TCPをトランスポートとして利用することができるため、TCPやUDPを利用して作成された既存のアプリケーションを、原則として書き換えることなく利用できるという大きなメリットが存在する。その反面、複数のアドレスはIPレイヤーでまとめて管理されることになるため、プロセスごとに適切なアドレスの組み合わせを設定することは容易ではない。

これに対して、本研究が採用したSCTPを用いて複数アドレスを活用する方法は、アプリケーションが利用するトランスポートを、SCTPへと変更する必要があるが、TCPからSCTPへの移行は、PIプログラミングスタイルを用いて作成されている場合には、アプリケーションを直接修正することなく、実現可能である。また、SCTPにおいては複数のアドレスが、プロセスごとに用意されたsctp_inpcb構造体へ直接格納されるため、プロセスごとにアドレス選択ポリシーして、適切なアドレスの組み合わせを容易に実現することができた。

sctp_inpcb構造体上のポリシーは、ソケットの作成時に環境変数からコピーされるものであるため、アプリケーションを起動した後に変更することができない。しかし、迂回路を利用する必要性がなくなった等、アドレス選択の基準が変化する場合があるため、アドレス選択ポリシーはアプリケーション起動中においても変更可能であることが好ましい。このような動的な変更を環境変数を利用して行うことは、環境変数がプロセスごとに管理され、外部から変更することが困難なものであることから適切ではない。動的な変更を実現するためには、sysctlコマンドのような外部コマンドを用いて変更を行うことができるようにする必要があると思われる。

7 おわりに

この研究では、SCTPによるアドレス自動更新モードを利用する際に取得したアドレスを用いるか否かのポリシーを事前に与えておくことで、SCTPを利用するアプリケーションがこのポリシーに基づきアドレスを取捨選択するようにする手法を提案した。

また、SCTPのADD-IP拡張によるアドレスの自動更新モードには、通信出来ないアドレスを利用しようとした際に無駄にタイムアウト待ちが必要となったり、最悪のケースでは通信が継続できない場合があることを示し、この問題に対処するために複数のASCONFチャンクをまとめて送出する手法を提案した。[1]その上で、これらの手法に基づく実装を行い、実験により提案手法が意図通りに動作することを確認した。

ところで、今回の実装においてはポリシーの設定に環境変数を用いる手法を採用した。この手法は既存のコードを修正することなく、利用者がプロセスの起動時にポリシーを設定できる利点があるがプロセスの実行中に、状況に合わせてポリシーを動的に変更するような細かい制御を行いにくいという欠点を持ちあわせる。そのため今後の拡張にあたってはsysctl関数などを用いたAPIによる手法で動的にポリシーを変更出来るような実装とす

[1] この手法はdraft-marushin-sctp-asconfext-01[12, 13]として標準化提案されたあとADD-IPに統合され、その後RFC5061[14]として公開されることとなった。

る方が望ましいと考えられる点をさらに検討していきたい。また今回の実装ではポリシーはユーザが決定するものとしたが、運用上の都合を考えると管理者が与えるポリシーとユーザが決定するポリシーの2種類が必要になるものと思われる。この点も踏まえて今後の拡張を行いたい。

なお本研究は科学研究費補助金基盤(B)、マイクロソフト産学連携研究機構戦略的支援プロジェクトならびにCisco URPの支援を受けている。また、Cisco Systems株式会社のRandall Stewart氏をはじめ，IETFにおけるSCTP研究チーム、またKAMEチーム等，多くのSCTP研究者に議論に参加して頂いた。この場をお借りして心よりの感謝を申し上げたい.

参考文献

1) R. Stewart, Q. Xie, K. Morneault, C. Sharp, H. J. Schwarzbauer, T. Taylor, I. Rytina, M. Kalla, L. Zhang and V. Paxson, "Stream Control Transmission Protocol", RFC2960, http://www.ietf.org/rfc/rfc2960.txt, October 2000.

2) R. Stewart, M. Ramalho, Q. Xie, M. Tuexen and P. Conrad, "Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", http://tools.ietf.org/html/draft-ietf-tsvwg-addip-sctp-15,

3) M. Tuexen, R. Stewart, P. Lei, E. Rescorla, "Authenticated Chunks for the Stream Control Transmission Protocol (SCTP)", http://www.ietf.org/rfc/rfc4895.txt,

4) R. Stewart, Q. Xie, L. Yarroll, K. Poon and M. Tuexen, "Sockets API Extensions for Stream Control Transmission Protocol (SCTP)" ,http://tools.ietf.org/html/draft-ietf-tsvwg-sctpsocket-15.

5) Kazu Yamamoto, "Protocol Independent Programming", http://www.mew.org/~kazu/doc/piprog.html, December 2003.

6) "SCTP Kernel Implementation",http://www.sctp.org/download.html.

7) D. Johnson, C. Perkins, J. Arkko, "Mobility Support in IPv6", RFC3775, http://www.ietf.org/rfc/rfc3775.txt, June 2004.

8) R. Wakikawa, T. Ernst, K. Nagami, V. Devarapalli, "Multiple Care-of Addresses Registration", http://tools.ietf.org/html/draft-ietf-monami6-multiplecoa-03, July 2007.

9) R. Moskowitz, P. Nikander, P. Jokela, T. Henderson, "Host Identity Protocol", http://tools.ietf.org/html/draft-ietf-hip-base-09, October 2007.

10) T. Henderson, "End-Host Mobility and Multihoming with the Host Identity Protocol", http://tools.ietf.org/html/draft-ietf-hip-mm-05, March 2007.

11) E. Nordmark, M. Bagnulo, "Shim6: Level 3 Multihoming Shim Protocol for IPv6", http://tools.ietf.org/html/draft-ietf-shim6-proto-08, May 2007.

12) 丸山伸, 小塚真啓, 中村素典, 岡部寿男, アドレス情報の変更通知を集約して再送できるようにするmSCTP拡張, 信学技報, Vol.106, No.173, IA2006-16, pp. 31-36

13) S. Maruyama and M. Kozuka, "Stream Control Transmission Protocol(SCTP) Cumulative ASCONF chunk transmission extension", http://tools.ietf.org/html/draft-marushin-sctp-asconfext-01, June 2006.

14) R. Stewart, Q. Xie, M. Tuexen, S. Maruyama and M. Kozuka, " Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration", RFC5061, http://www.ietf.org/rfc/rfc5061.txt, September 2007.