第 XIII 部

SCTP および DCCP に関する研究開発

第 13 部
SCTP および DCCP に関する研究開発

第 1 章　はじめに

SCTP ワーキンググループは、次世代のトランスポートプロトコル：SCTP（Stream Control Transmission Protocol）や DCCP（Datagram Congestion Control Protocol）などに関する研究活動を行っている。本年度の主な活動内容は以下の通りである。

- WIDE 合宿を利用した実証実験の総括
- SCTP Profiling Module の設計と実装
- SCTP と Mobile IPv6 を組み合わせた効率的なハンドオーバ手法の検討
- SCTP と ADD-IP による高速ハンドオーバ手法の研究

以下、各項において詳細な内容を報告する。

第 2 章　WIDE 合宿を利用した実証実験

2.1 実証実験の目的と結果の概要

SCTP ワーキンググループにおいては、ワーキンググループ設立以来 2 年間にわたり SCTP（Session Control Stream Protocol）[236] に関する調査研究や機能拡張を行ってきた。同時に、各種 OS に対する SCTP の実装においても主導的に活動してきた。

これらの実装の動作を確認しつつ多くの人に実際に SCTP を利用してもらうことにより、現在の仕様や実装の問題点を洗い出すことを目的として、2006 年 3 月および 9 月に開催された WIDE 合宿において実証実験を行った。

1 回目の実験は 2006 年 3 月に行った。実験環境の準備の過程において、現行の SCTP 実装には事前の想定以上に多くの問題点があることが確認された。実際に実験を行ったことでさらに多くの問題点が確認された。

この問題点の多くは、連続してアドレス変更が発生した際に、通信相手にアドレス変更通知をおこなう ADD-IP 拡張の問題であった。

そのため、この実験により認識された問題点を改良するために “Cumulative ASCONF” と呼ばれる拡張仕様 [235] を考案し、7 月にモントリオールで開催された IETF-66 において提案した。

その後、8 月末には京都府京丹後市網野町において 1 週間にわたり合宿を行いこの実装を集中的に行なった。

その結果、2006 年 9 月に開催された WIDE 合宿において 2 回目の実験を行い、SCTP を用いたモビリティを多くの参加者に経験してもらうに至った。

2.2 実験の内容

WIDE 合宿という場を利用し多数の参加者の協力を得て、SCTP ハンドオーバ実証実験を行った。

SCTP を利用するサービスやアプリケーションがまだ本格的に存在しない現状でより多くの参加者に SCTP を利用した実験に参加してもらうために、参加者の利用するクライアント側ではライブラリ実装による SCTP を用いることにした。これにより SCTP 非対応の OS 環境においても SCTP を利用できるようになる。また SOCKS による通信を SCTP 経由で中継するような特殊な SOCKS デーモンを開発することで、SOCKS 対応のアプリケーションであれば SCTP を利用した IP ハンドオーバーを行えるようにした。この SCTP のライブラリ実装（SCTPlib）は FreeBSD、Mac OS X、Linux のみならず、Windows でも動作するようにしたことで、ほとんどの端末で本実験に参加できるようになった。

サーバ側では FreeBSD 6 系列が動作するサーバに SCTP 対応パッチを適用した上で、SCTP による接続を受け取り TCP によって目的とするサービスに接続を行う SOCKS デーモンを開発した。

このような準備により SCTP ハンドオーバ実験に多数の参加者を得ることができ、SCTP の実運用において顕在化する様々な問題点やデータを収集することができた。

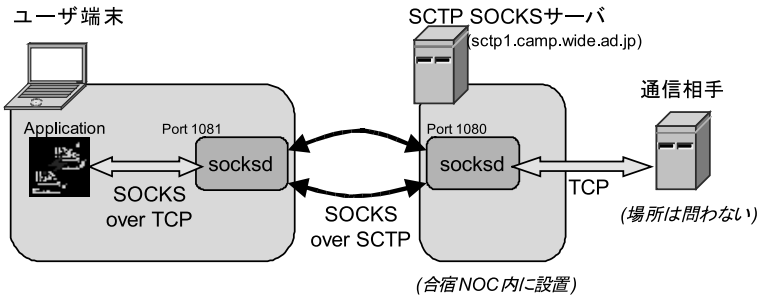

図 2.1. 実験構成

2.3 問題点とその対策

多くの参加者を得て実験を行ったところ、仕様の想定した通りには動作しない現象がいくつも確認された。以下、実験を通じて顕在化した問題点の詳細を述べると共に、それらに対して行った対策についても述べる。

2.3.1 SCTP のライブラリ実装（SCTPlib）の問題

SCTP のライブラリ実装（SCTPlib）には Ver1.0.X 系列と Ver1.3.X 系列の 2 つの実装が存在している。但し、SCTPlib の開発は、主力開発者であった Michel Tuexen が Mac OS X 上での SCTP 開発にシフトしたことにより、実質停止していた。すなわち、Ver1.0.X 系列は、Windows を含め様々な OS で動作するが、ADD-IP 拡張が実装されておらず、Ver1.3.X 系列は、ADD-IP 拡張が実装されていたが、UNIX 系 OS 上でのみ動作するライブラリとなっていた。

多くの合宿参加者に実験に参加してもらうため、様々な OS（特に Windows）において SCTPlib が動作することが重要であったことから、Ver1.0.X 系列をベースとして、ADD-IP 拡張を実装した（2006 年春の WIDE 合宿当時）。

その後、FreeBSD 上の SCTP カーネル実装において、ADD-IP 拡張に AUTH 拡張が必須となったため、春の合宿で用いた SCTPlib に AUTH 拡張を実装した。また、春の合宿で判明した問題を回避するために、IETF-66 において提案した、Cumulative ASCONF（2.4 節参照）も実装した（2006 年秋の WIDE 合宿当時）。

2.3.2 ソースアドレスセレクション問題

SCTP では HeartBeat チャンクや ASCONF チャンクに対する応答は、元のチャンクが含まれていたパケットのソースアドレスに送り返すと規定されている。これらのチャンクを送出する際には、適切なソースアドレスを選択する必要がある。また、パケット送出に用いるソースアドレスは、アソシエーションに含まれているものでなければならない。

オリジナルの SCTPlib はパケット送出時のソースアドレス選択を OS に任せていたが、この手法では、アソシエーションに含まれていないソースアドレスを用いたり、到達性のないアドレスをソースアドレスに設定して、パケットを送出してしまう可能性があった。そこで、SCTPlib においてソースアドレスを選択するように実装を変更した。具体的には、ASCONF チャンクを送出するパケットのソースアドレスは、新たに追加したアドレスを優先的に用いる。また、それ以外のチャンクについては、同一のデスティネーションアドレスに対して、ソースアドレスを変えて HeartBeart チャンクを送出し、HeartBeat ACK による応答があったソースアドレスを用いている。

2.3.3 出力インタフェイス問題

端末に複数のインタフェイスが接続され、それぞれのインタフェイスに異なる IP アドレスが付与されている場合においても、SCTPlib はパケット送信の際に、インタフェイスを使い分けることが出来ない。このため、例えば無線 LAN と PHS を同時に利用する場合には、データ送信は常に低速な PHS を用いるといった非効率が発生する。

この問題を解決するために、我々はソースアドレスに基づいた経路選択を提案しており、次回の実証実験において検証する予定である。

2.3.4 アドレス切り替え検出の遅延

Windows 標準の無線 LAN ユーティリティを用いると、部屋の移動等によって、無線 LAN によって接

続するネットワークが変化した場合においても、移動前のネットワークに接続していた時に利用していたアドレスが利用できるかどうかの確認を行った上で、古いアドレスを削除した上で、新たなアドレスを取得するという処理が行われる。SCTPlib は、無線 LAN インタフェイスを直接監視することなく、アドレスの変化のみを検出してハンドオーバを行う実装となっていたため、接続先ネットワークの切り替わってもしばらくはハンドオーバが完了しなかった。

2.3.5 ハンドオーバー完了後のデータ送信の遅延

ASCONF チャンク等の交換によって、ハンドオーバが完了した後であっても、データ送信にあたってハンドオーバを考慮していないため、データ送信の再開に極めて長い時間を要する現象が多く見られた。

この問題は、再送アルゴリズム等のハンドオーバ対応によって改善されると考えられ、次回の実証実験において検証を予定している。

2.3.6 複数のアドレスが次々と増減することにより生じる問題

この問題の詳細と我々の行った提案については、次節において詳論する。

2.4 Cumulative ASCONF の提案

上述した問題点のうちのいくつかは、従来の ADD-IP ドラフトにおける「同時に 1 つの ASCONF チャンクしか送出してはいけない」という制約に基づくものであった。この制約があるために、アドレスの増減が発生した場合などに、変更を通知するための ASCONF チャンクが送出された後、相手によって適切に処理された上で ASCONF-ACK チャンクとして返送されなかった場合、それ以降にアドレスの増減が発生したとしても新たな ASCONF チャンクを送出できないことに起因する。

この制約はプロトコルをシンプルに保つことを目的としているのと同時に、ASCONF チャンクにより Flooding を引き起こすことがないようにとの配慮によるものであった。そこで、以下のような拡張を行うことで従来の条件を変更することなく ASCONF を拡張し、複数の ASCONF を同時に送れるようにする手法を提案した [312]。

- 未解決の ASCONF がある時に新たな ASCONF を送る必要がある時には、それらの ASCONF を全て束ねて（Bundle）送るようにする
- 複数の ASCONF が束ねて送られてきた時にはそれらを順次処理する

この提案は Internet-Draft “Cumulative ASCONF chunk transmission extension”[136] としてまとめられ公開されているので、詳細はこちらを参照して頂きたい。

次いで 2006 年 9 月に開催された WIDE 合宿における 2 回目の実験においては、この提案に従った実装を行った SOCKS を利用することで、アドレス変更時の安定性が大幅に向上することが確認された。また、この提案は ADD-IP 拡張を定義している Internet-Draft に取り込まれ、“Stream Control Transmission Protocol (SCTP) Dynamic Address”[237] として公開されている。

2.5 今後の課題

SCTP を用いた IP ハンドオーバを行う際には、インタフェイスが 1 枚の状況も十分に考慮しておく必要がある。しかしながら、このときのアドレス変更には旧アドレスの切断判定から新アドレスの割当てに至るまでに非常に長い時間が必要であることが確認されている [310, 311]。この問題に対しては SCTP のみの枠組で考えようとするのではなく、他のレイヤとの情報交換なども通じて解決への道を探っていきたい。

2.6 まとめ

SCTP ワーキンググループでは 2 回の実験を通じて実運用を行う際に発生する様々な問題を解決する手法 “Cumulative-ASCONF” を提案した。この手法を実装し実験を通じて効果を確認すると同時に、IETF において標準化するべく活動している。今後はさらなる研究や実験を行うと同時に必要な拡張を行うことで、SCTP の普及にむけての貢献を行いたい。

第 3 章　SCTP Profiling Module の設計と実装

Abstract

Nowadays, it is not rare that a mobile node equips multiple interfaces such as Ethernet,

802.11b, mobile phones and WiMAX. A node with multiple interfaces has the possibility to utilize the multiple network accesses for robust and efficient communication. However, TCP can handle only a single IP address per connection. Hence, applications that uses TCP can not utilize multiple network access fully. The Stream Control Transmission Protocol (SCTP) is a new transport protocol which is receiving attention as the successor of TCP. One of the advanced features of SCTP is multi-homing, which allows an endpoint to have more than one IP address for switchovers to be transparent to higher layers. In this document, we focus on issues in SCTP multi-homing and propose a new framework that enhances this feature to realize smooth switchover. By estimating the condition of each networks, such as delay or loss-rate, our framework enables SCTP to use the best address that meets users' requirements. Since our framework will be useful for delay-sensitive or loss-rate sensitive applications in multi-homed environments, it will be suitable for applications in mobile environments.

3.1 SCTP Multihoming and Switchover

3.1.1 Multihoming Mechanism in SCTP

A host with multiple network interfaces can be assigned multiple IP addresses on an IP network. These kinds of hosts are referred to as "multi-homed" hosts[21]. Multi-homing is a technique used to increase the reliability of the network connection in multi-homed environments. The concept of multi-homing is simple. If the reachability of one IP address is lost, other IP addresses will be used to maintain network connectivity of the host. However, applications which use TCP cannot benefit by this technique. This is because a TCP endpoint can only support a single IP address per connection. Hence, when a host decides to change the IP address in use, all TCP connections on the hosts will be terminated and have to be reconnected. This will cause extra overhead to applications and network traffic since it might force some applications to start over. On the contrary, a SCTP endpoint can support multiple IP addresses per association. This feature allows SCTP to change the IP address for communication without terminating.

3.1.2 Switchover Mechanism in SCTP

As mentioned above, an SCTP application on a multi-homed host can increase the stability of communication by utilizing multiple IP addresses on the host. During association setup, each SCTP endpoint exchanges the list of available IP addresses on the host. Each endpoint selects a pair of destination and source addresses from the address list and defines it as the primary path. SCTP sends all data through this primary path for normal data transmission. If SCTP fails to receive an acknowledgment for a data packet before the retransmission timeout (RTO), it increases error count by one and retransmits the data packet to another destination address in the peer's list. If the sender SCTP receives an acknowledgment for the retransmitted packet, further data is transmitted to the primary path again.

To verify the reachability of the peer, SCTP transmits heartbeat packets to all destination addresses which include addresses marked inactive. The heartbeat packets are special packets which are transmitted at constant intervals. If an address acknowledges the heartbeat, it is assumed that the address is reachable. Otherwise, error count for the address is increased until it exceeds the threshold and the address is marked inactive. The default value of the error count, *Path.Max.Retrans* is 5, as recommended in the RFC[236].

3.1.3 Issues in SCTP Switchover

First issue in SCTP switchover is that the failing over process takes time. A sender tries to retransmit a data packet 5 times until it considers the primary path to be inactive and switchover to another path. This is because *Path.Max.Retrans* is set to 5 as recommended. The retransmission

timeout is doubled each time a timeout occurs. Thus, it takes over 30 seconds from first packet lost to the time that the primary path is marked inactive and another path is chosen as the new primary destination. Since only several packets have been transmitted during switchover process, this is critical for communication performance.

We might be able to improve this behavior by reducing *Path.Max.Retrans* to a lower value, such as 3. In this case, the switchover time will be reduced to about 7 seconds, which might be better but still causes great performance degradation. Further reduction of *Path.Max.Retrans* might introduce a ping-pong effect where the address used switches from one to another too frequently that it might cause problems. The second issue is that switchover is triggered only when reachability of the path is completely lost. If the primary destination does not fail completely, but it becomes limited due to certain factors (for example an increase in delay or a decrease in bandwidth), SCTP does not recognize this change and continues to use the primary destination.

3.2 Profiling Framework for SCTP

To solve the problems presented in the previous subsection, we propose the SCTP function that measures the characteristic of all possible paths and ranks the paths based on these measurements. The SCTP specification provides a special chunk type called heartbeat used for checking the reachability of the available destination addresses. Our proposal uses this heartbeat chunk to measure the characteristic of the destination. We also provide a special API that allows applications to specify preferable characteristics for its path. Our scheme tries to provide the best path for applications based on the rank of the paths. For example, if an application specifies small delay preference, our scheme always chooses the path with the smallest delay for the primary path. Since our scheme allows SCTP to switchover to another destination address without waiting for multiple timeouts, it can improve communication performance drastically when some errors occur on the primary path. Since sending heartbeat chunks conforms to SCTP specifications, our proposal does not violate the SCTP standard. In addition, our proposal does not require any modification on data receiver side.

3.3 Implementation

We implemented our scheme in the SCTP reference implementation on FreeBSD5.4. We use the SCTP code from the SCTP Kernel Implementation for FreeBSD[214].

Figure 3.1 presents a graphical overview of the architecture for our implementation. The main part of our implementation is the *Profiling Framework* that provides the necessary functionality to the SCTP stack in the kernel. The *Profiler Modules* implements different algorithms to measure the characteristic of the path and can be used to offer different switchover services to user applications. Since the Profiler Modules are implemented as loadable kernel modules, system administrators or users can load or unload these modules for their own purpose. Applications that want to use our Profiling Framework can specify a preferred policy through APIs. The Profiler Modules which are specified in applications should be loaded before the applications are executed.

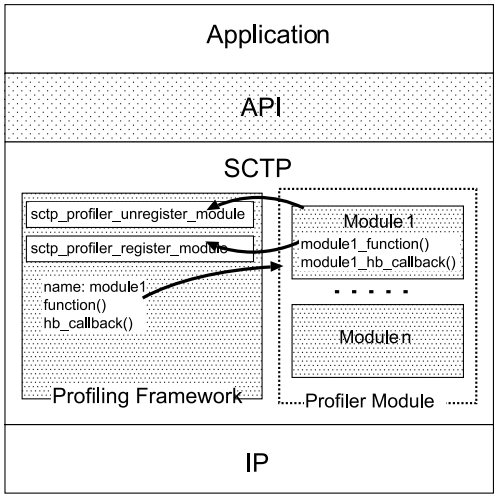

Fig. 3.1. Architecture of the Profiling Framework for SCTP

3.3.1 Profiling Framework

The Profiling Framework is the function that manages all Profiler Modules and associates them with user applications. When the Profiler Modules are loaded, the Profiling Framework registers the modules and starts to maintain the status of the modules. When a module is used by applications, the Profiling Framework invokes the functions in the module accordingly. To register or unregister Profiler Modules, the Profiling Framework supports two functions: *sctp_profiler_register_module()* and *sctp_profiler_unregister_module()*. These two functions are called from the Profiler Modules when they are loaded or unloaded.

3.3.2 Profiler Modules

As stated above, each SCTP Profiler Module implements different algorithms to measure the characteristic of the path and provides its own switchover service to applications. The SCTP Profiler Modules are implemented as loadable kernel modules so that they can be unloaded when they are not needed. A loadable kernel module is easier to develop and debug than a built-in implementation in the kernel.

Each Profiler Module needs to register three types of information: *name*, *function*, *hb_callback* to the Profiling Framework when it is loaded. The *name* type is the unique identifier of the module. This identifier is used for applications to specify the modules which they want to use. The *function* type is the pointer to the function in the module which is invoked periodically from the Profiler Framework. The *function* type manages heartbeat packets transmission based on its algorithm and calculates measurement results and switches the primary destination when it is necessary. The *hb_callback* type is also a pointer to the function in the module which is invoked from Profiler Framework whenever a heartbeat acknowledgment (HB ACK) arrives. It is used to update the measurement results by analyzing information in HB ACKs.

3.3.3 API

Application programmers can choose their preferred profiler function through an API. For this purpose, we define a special structure: *struct sctp_profiler_opt*. This structure is used for a *setsockopt()* or *getsockopt()* call with the special optname: *SCTP_PROFILER_OPT*. Figure 3.2 shows the definition of the *sctp_profiler_opt* structure. The *sprof_modulename* is the name of the module that is to be used. This value should be one of the module names registered in the Profiling Framework. The *sprof_assoc_id* is the identifier of the SCTP association which should be associated with the module. If this value is set to 0, it means that this module should be used as the default for all new associations. The *sprof_interval* specifies the interval that the *function* in the module is called. If this value is set to 0, the module will use a pre-defined default value for the interval. If this value is set to −1, it means that the module should be deactivated for the specified association. Figure 3.3 shows an example usage of the Profiler Framework. In this example, we specify the use of the RTT profiler modules which will be explained later. This module will be applied to all new associations, since *sprof_assoc_id* is set

```
struct sctp_profiler_opt {
        sctp_assoc_t sprof_assoc_id;
        char sprof_modulename[256];
        u_int32_t sprof_interval;
};
```

Fig. 3.2. sctp_profiler_opt structure

```
struct sctp_profiler_opt opts;

  opts.sprof_assoc_id = 0;
  strcpy(opts.sprof_modulename,
     ''profiler_rtt'');
  opts.sprof_interval = 1000;
  setsockopt(sockd, IPPROTO_SCTP,
     SCTP_PROFILER_OPT, &opts, sizeof(opts));
```

Fig. 3.3. An example of an RTT profiler module

to 0. The measurement interval is set to 1000 millisecond.

3.4 Conclusion and Future Work

In this document, we discussed the issues in the switchover mechanism in SCTP and proposed a new framework that enhances this feature for performance improvement. We provide handy APIs so that existing SCTP applications can utilize our framework by just adding a few lines of code. We have developed simple Profiler Modules: m_lossprob and m_rtt and examined that our modules enable applications to switch the primary path immediately without waiting for consecutive timeouts. To evaluate our method, we conducted various tests in our experimental networks and confirmed that our modules can switchover to alternate path within 2 seconds while original SCTP takes over 30 seconds for switchover. Our framework will be especially useful especially for mobile nodes that equip multiple wireless interfaces. Although the network condition of wireless networks changes frequently, our framework can hide it by choosing the best path among the available paths. Hence, delay-sensitive or loss-rate sensitive applications can benefit from our framework. As future work, we plan to develop a streaming application for mobile devices based on our framework and evaluate the performance of our system.

第 4 章　SCTP と Mobile IPv6 を組み合わせた効率的なハンドオーバ手法の検討

4.1 Introduction

Mobility support is the key feature to realize all IP mobile networks. Many mobility technologies have already been introduced and standardized at the Internet Engineering Task Force (IETF). Mobile IPv6[112] is a mobility protocols which enables a mobile node to roam between subnetworks without any session break. However, during handover, the small break can be observed because the mobile node must update its binding to newly acquired care-of address. To do so, the mobile node sends a binding update to its home agent and may also transmits binding updates to correspondent nodes. Before completion of binding updates, the mobile node cannot communicate with any nodes. The latency depends on round trip time between a mobile node and either a home agent or a correspondent node. This handover latency must be minimized so that users and applications are unaware of roaming period.

It is expected that several new services will be built on all IP mobile networks, such as 4G cell phones, vehicle communications (Intelligent Transport System) and Personal Area Network. Real-time applications such as VoIP and streaming are sensitive to even small session break. Although it is required that the handover latency must be minimized, Mobile IPv6 does not provide optimized scheme for smooth handover. Thus, several researches are introduced to minimize the handover latency by extending Mobile IPv6 protocol such as FMIP. FMIP will be useful when a mobile node equips with single wireless interface. However, a mobile node now has many wireless interfaces to access the Internet and can handle multiple interfaces simultaneously. There are also some proposals to manage multiple interfaces and IP addresses on Mobile IPv6[265]. While handover operation is proceeded, one of active interface can accommodate traffic on behalf of roaming interface. In addition, a mobile node may utilize multiple interfaces simultaneously for load balancing and bandwidth aggregation. In future, it is obvious that these technologies are operated on all IP mobile networks, but the modifications required by these technology are spread to everywhere in Mobile IPv6: a mobile node, a correspondent node, and a home agent.

We investigate the use of Stream Control Transmission Protocol (SCTP)[236] to achieve low handover latency on regular Mobile IPv6. SCTP is capable of handling multiple addresses for

a session and has a feature of failover mechanism when one of address is failed. The home address is used for correspondent nodes to identify the mobile node all the time. Our scheme is designed not to modify existing protocols at all. Only operations of a mobile node are newly defined between SCTP and Mobile IPv6.

4.2 Mobile IPv6 and SCTP

In this section, we briefly describe Mobile IPv6 and SCTP.

4.2.1 Mobile IPv6

Mobile IPv6 allows a mobile node to be addressed by a home address all the time even though the mobile node changes its point of attachment to the Internet. When the mobile node is attached to a new network, the mobile node sends a binding update to its home agent, a router on the mobile node's home link. A binding update describes the relation between the home address and an IP address associated with the mobile node while it is on the visiting link, called care-of address. When a correspondent node sends a packet to the home address of the mobile node, the home agent receives the packet by normal routing in the Internet. Since the home agent has the binding for mobile node, the home agent can forward the packet to the current mobile node's care-of address by the bi-directional tunnel. After the mobile node receives the tunneled packet, the mobile node sends a binding update, causing the correspondent node to cache the mobile node's binding into its binding cache database. Consequently, the correspondent node routes packets directly to the mobile node's care-of address according to the registered binding cache.

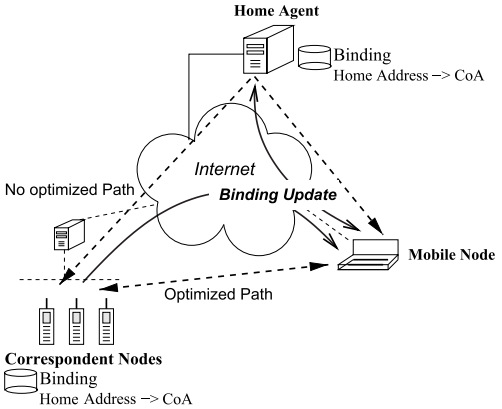

Fig. 4.1. Mobile IPv6

4.2.2 SCTP

Stream Control Transmission Protocol (SCTP)[236] is a newly defined transport protocol. The feature of SCTP is message oriented reliable transmission while TCP is byte-oriented. The message oriented transmission is suitable for telephony applications which often required rigid timing. TCP provides reliable transmission, but the rigid timing is hard to make because of strict TCP packets' sequence management. In addition, TCP does not provide redundancy of transmission when an IP address becomes invalid, while SCTP supports multihomed IP communication. The mobile SCTP is introduced to support seamless mobility by using SCTP multihomed IP support feature[200]. The advantage of mobile SCTP is that no modification is needed in networks, while Mobile IPv6 requires a home agent to provide mobility support. However, the mobile SCTP does not provide reachability assurance because of address changes. To support correspondent node initiated communication, another mechanism such as Dynamic DNS is needed to reach the mobile node.

4.3 System Overview

Figure 4.2 shows our system overview. A mobile node has multiple interfaces in order to access the Internet permanently. This configuration is quite common these days even for small terminals like mobile phones. The mobile node obtains multiple IP addresses at the visiting network at each interface. It also run Mobile IPv6 and manage a home address as its unique identifier. The mobile node selects one of IP address as a care-of address of Mobile IPv6 to send a binding update. Once the

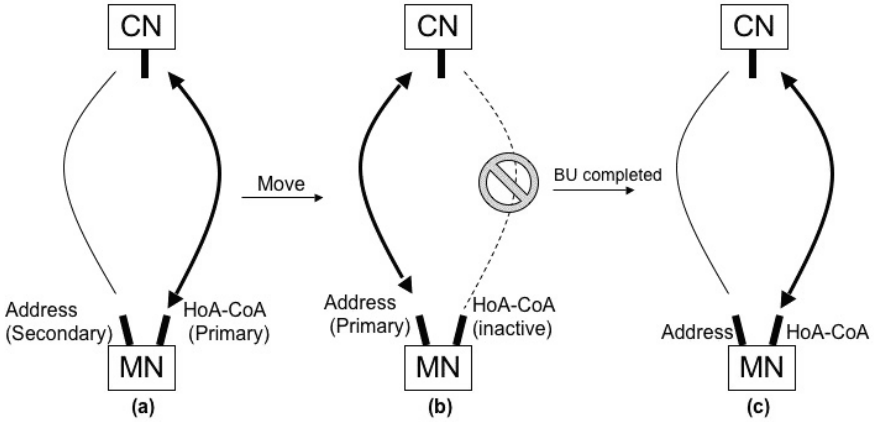

Fig. 4.2. System Overview

mobile node completes binding registration, it can start SCTP session at the home address. The home address becomes a primary address in the SCTP session. During the communication, the mobile node notifies the IP addresses other than the care-of address as secondary IP addresses to correspondent nodes (Figure 4.2-a).

The secondary IP addresses will be used only when the primary IP address becomes invalid due to handover. This failover is conducted by SCTP (Figure 4.2-b). After all the binding updates are completed, the mobile node notifies correspondent node to let the home address be a primary address again. Then, the communication is now back to the primary address which is home address (Figure 4.2-c). Note that our scheme assumes that the mobile node always uses only one of interface for communication. This limitation is from SCTP specification, because SCTP uses multiple addresses management feature only for redundancy purpose.

The possible issue of SCTP is that the failover operation takes certain time. The SCTP specification[236] stated that the failover will be begun when the retransmission timeout is occurred *Path.MAX.Retrans* times which default value is 5. If the initial timeout value is set to 1 sec, the average time for the SCTP failover procedure is about 30 seconds which is too slow. However, this can be optimized to tune SCTP parameters and to use other layer information such as link status events to start SCTP failover process.

4.4 Evaluation

The advantage of our scheme compared to other smooth handover mechanisms are listed below.

- When the fail over is occurred, the trigger of flow redirection is done by SCTP. Since IP does not have such capability in original, it is more natural to handle it in transport layer.
- End nodes do not need to extend Mobile IPv6 and SCTP, while most of related work extend Mobile IPv6.
- During the movement procedures, a mobile node can send and receive packets at one of active interface. Once it finishes binding registrations, the traffic is redirected to the home address.

We developed Mobile IPv6 stack on BSD operating system, called SHISA. Since BSD supports SCTP in the kernel, SHISA is used to evaluate our scheme. We conduct experiment running Mobile IPv6 and SCTP at the same time and investigate how this scheme is effective for smooth handover. The handover latency is compared among our scheme, original Mobile IPv6, and multiple care-of address registration ([265]).

4.5 Conclusion

This document presents a smooth handover scheme for Mobile IPv6 by using SCTP failover

mechanism. The scheme does not require any modification to existing protocols, though it may need some parameter tuning to enhance smooth handover. There are several mechanisms to minimize handover latency, but most of proposals need certain amount of modification to end nodes and home agents. From the view of short period, modification to protocols on a number of nodes are not realistic work on the Internet. Therefore, our scheme can be useful to achieve low handover latency to non-modified end nodes.

第 5 章　SCTP と ADD-IP による高速ハンドオーバ手法の研究

SCTP と ADD-IP によるハンドオーバは、中継ノードに依存しないことが利点として挙げられる。しかし現在の SCTP と ADD-IP の仕様では単一のインタフェースでハンドオーバを行う際に、宛先毎の輻輳制御や、ADD-IP の機能の一つである、通信相手のプライマリアドレスを指定する機能に問題がある。本研究では送信元と宛先の組み合わせごとに輻輳制御を行うこと、ADD-IP の機能を一部変更することで、単一のインタフェース上における高速なハンドオーバを実現した。

第 6 章　その他の活動

SCTP ワーキンググループでは上記の活動に加え以下の活動も行った。それぞれの活動について概要を報告する。

6.1 Randall Stewart 氏とのミーティング

6 月と 10 月に、Cisco Systems の Randall Stewart 氏が来日し、京都大学において SCTP 研究に関するミーティングを行った。このミーティングでは、Stewart 氏と 2006 年春の WIDE 合宿の実証実験によって判明した ADD-IP 拡張の問題点の議論が行われ、我々はこの議論を基として ADD-IP 拡張に対する修正を第 66 回 IETF において提案した。我々が提案した ADD-IP 拡張に対する修正は、ADD-IP 拡張に取り込まれ、現在標準化作業中である。

6.2 Lars Eggert 氏とのミーティング

10 月 18 日に IETF Transport Area の Area Director である Lars Eggert 氏と慶應義塾大学三田キャンパスにおいてトランスポートプロトコル研究に関するワークショップを開催した。WIDE プロジェクト側からは 10 数名が参加し、WIDE プロジェクトの紹介や各メンバが現在取り組んでいる研究に関するプレゼンテーション、Eggert 氏からは、TCP User Time Stamp Option などに関する解説が行われた。いずれの発表でも活発な意見交換が行われた。

第 7 章　おわりに

本年度の SCTP ワーキンググループの活動では、SCTP 実装の開発を進めると同時に ADD-IP の拡張など、IETF における標準化活動にも貢献することができた。また WIDE 合宿の実験結果より、新たな研究テーマが生まれ、活発な研究活動を行うこともできた。来年度は、新たな人材の発掘を行い、現在の研究テーマの発展と新規テーマの検討を進めていく予定である。