TMSP: Terminal Mobility Support Protocol

Teck Meng Lim, Chai Kiat Yeo, Bu Sung Lee, Quang Vinh Le

Abstract— **Mobile IP enables IP mobility support for mobile node (MN), but it suffers from triangular routing, packet redirecting, increase in IP header size and the need for new infrastructure support. This paper details an alternative to enable terminal mobility support for MN. This scheme does not suffer from triangular routing effect and does not require dedicated infrastructure support such as home agent. It also does not increase the size of the IP header and does not require redirection of packets. These benefits are enabled with a tradeoff which requires modifications on MN and its correspondent node. It uses an innovative IP-to-IP address mapping method to provide IP address transparency for applications and taps on the pervasiveness of SIP as a location service. From our analysis, we show that TMSP is much more efficient than mobile IP in terms of the number of hops as well as overhead. Our prototype implementation also shows that TMSP provides seamless communication for both TCP and UDP connections and the computational overhead for TMSP has minimal impact on packet transmission.**

I. Introduction

With the rapid deployment of session initialization protocol (SIP) [1], the number of VoIP users has grown rapidly in recent years. The traditional SIP only provides personal mobility whereby it allows a single user to be located at different terminals and referenced by the same logical SIP uniform resource identifier (URI). However, once a connection is established, the movement of this user which results in a change in its terminal's IP address will lead to a loss in the connection. This restricts the mobility of WiFi VoIP phones and PDAs within a network once a session has been established. With recent progress of wireless data network which provides many options for mobile connectivity, like IEEE 802.11 (WLANs), universal mobile telecommunication system (UMTS), IEEE 802.16 (WiMAX), and many other competing standards, with overlapping coverage, terminal mobility will become essential for these devices to maintain seamless sessions as they move across these data networks.

The IETF proposes mobile IP for IPv4 (MIPv4) [2] and IPv6 (MIPv6) [3] to enable terminal mobility. However, it introduces a triangular routing phenomenon or packet redirection which causes sub-optimal effects like longer route leading to increased delay, and susceptibility to link failure, additional infrastructure load, and increased packet overhead leading to increased processing delay on terminals and increased chances of packet fragmentation.

This work is supported by the Agency for Science and Technology Research (A*STAR) of Singapore. (Project number: 0421010012 - M47020012)

Teck Meng Lim is from StarHub Ltd, Singapore. Yeo Kiat Yeo and Bu Sung Lee are from the Centre for Multimedia and Network Technology, School of Computer Engineering, Nanyang Technological University, Singapore. Quang Vinh Le is from Fiserv. (e-mail: limteckmeng@gmail.com, asckyeo@ntu.edu.sg, ebslee@ntu.edu.sg, quangvinh@gmail.com)

We propose a novel mobility management method, terminal mobility support protocol (TMSP), that resolves IP mobility for mobile node (MN). Compared to MIPv4, TMSP provides a new engineering alternative for route optimization with the need to modify both the MN and its correspondent node (CN) instead of modifying only the MN in MIPv4. It ensures seamless data connection between a MN and its CN even when MN crosses wireless networks. CN is a peer with which a MN is communicating and can either be connected to the Internet via wired or wireless networks. It does not rely on new infrastructure support, like mobile IP, which requires a home agent (HA). HA is a special network server to provide redirection of packets for MN based on a binding of MN's permanent IP address, i.e. home address (HoA) to MN's current IP address, i.e. its care-of-address (CoA). A CoA is acquired by MN whenever it enters a foreign network. Unlike MIPv6, TMSP uses the pervasiveness of SIP services to provide an URI-to-IP address mapping to locate user, and a module installed on MN and CN to ensure transparency in IP address changes for data connections. Thus, users can use any mobile device and starts roaming using a URI.

TMSP possesses six major features. It 1) does not assign a permanent IP address to each MN, 2) uses a single IP address at any time for each network interface, 3) does not introduce new network servers/infrastructure support to enable IP mobility support, 4) does not need IP packets redirection (i.e. packets are routed directly between MN and its CN), 5) does not incur extra IP header extension on each packet, and 6) allows IPSec without rerunning Internet key exchange (IKE) [4].

We begin with a brief description of MIPv4, MIPv6 as well as application-layer mobility management for SIP applications proposed by Schulzrinne et. al [5], and identify the problem in using IPSec on MN. In section III, we describe the functional modules used in MN and the states kept in MN to facilitate mobility management. After that, we detail the operation of TMSP in boot strapping, connection establishment, packet pathway and signaling. We also study the advantages of using TMSP in section IV and present our implementation results in section VI. We discuss the related works on IP mobility in relation to TMSP and deliberate the issues on micro-mobility, firewall, NAT traversal, and session border controller and how TMSP can address such issues to support terminal mobility in section V. Lastly, before we conclude, we reiterate the motivations and discuss the limitations of TMSP.

II. Background

This section briefly introduces mobile IP and an application-layer mobility management proposal for SIP applications. It also outlines the main components and functions of IPSec, and identifies the problems of using IPSec on MN.

A. Mobile IPv4

MIPv4 specifies protocol enhancements that allow transparent routing of packets to MNs as they move their network attachment point in the Internet. Each MN is assigned a HA which resides in the home network of MN. Each MN is also assigned a HoA, regardless of its current network attachment point to the Internet and acquires a CoA whenever it enters a foreign network. MIPv4 provides a message exchange sequence called binding update for registering this CoA with the HA of MN so that HA creates a binding of MN's HoA to this CoA. Every packet destined to the MN reaches the HA since the destination IP address of all packets are set to MN's HoA. The HA then redirects or tunnels these packets to MN through an IP-in-IP tunnel [6]. After arriving at the end of the tunnel, each packet is then delivered to MN. In the reverse direction, packets sent by MN are delivered to CN directly without passing through MN's HA. This indirect route for packets is known as triangular routing.

MIPv4 provides two methods of acquiring a CoA. A "foreign agent CoA" is a CoA provided by a foreign agent (FA) through its agent advertisement messages. In this method, the FA is the end-point of the tunnel and, upon receiving tunneled packets, removes the tunneling encapsulation and delivers the inner packet to MN. A "co-located CoA" is a CoA acquired by MN as a current IP address through some external means like dynamic host configuration protocol (DHCP), which MN then associates it with one of its own network interfaces. In this case, MN serves as the end-point of the tunnel and it removes the tunneling encapsulation of the tunneled packets. Thus, using the co-located CoA, MIPv4 can operate without a FA.

B. Mobile IPv6

MIPv6 suggests two operation modes, basic mode and route optimized mode. The basic mode operates like MIPv4, except that it only uses "co-located CoA", thus FA is not required. It also uses a bidirectional tunnel between MN and HA. Thus, packets sent by MN will be tunneled to HA where they are decapsulated and sent to CN.

The route optimized mode avoids triangular routing by allowing packets from CN to be routed directly to the CoA of MN and vice versa. Here, it also uses mobility extension headers to reduce the effect of tunneling whereby packets are routed directly between CN and MN without the need for redirection by HA once a binding update from MN to CN is completed after a connection is established. It defines a new routing header variant, type 2 routing header, on each packet sent from CN to MN. This header contains the HoA of MN. Once a packet arrives, MN retrieves its HoA from the routing header, and uses it as the final destination address for the packet. Likewise, a home address option in the IPv6 mobility header is used to inform CN of MN's HoA.

C. Application layer mobility management proposal for SIP application

Application layer mobility using SIP has been discussed by Schulzrinne et al. [5]. This proposal works only for SIP applications. It uses the SIP invite request message to re-invite CN when a MN changes its IP address. After this invitation, the SIP application at CN will use MN's new IP address for subsequent packets to MN. However, this method only supports user datagram protocol (UDP) connections and the authors propose using mobile IP for SIP applications that require transmission control protocol (TCP) connections.

D. IPSec on IP Mobility

The IPSec protocol adds security to IP by enabling the sending and receiving of cryptographically protected packets. It is a mechanism used to provide packet level security. Each packet has a special IPSec header which is used to identify the types of cryptographic protection that are applied to the packet and also other information necessary for decoding the packet. IPSec has the following headers: authentication header (AH) [7] and encapsulated security payload extension header (ESP) [8]. The AH protects against malicious modification of a packet without providing privacy. It guarantees connectionless integrity and data origin authentication of packets. Optionally, it protects against replay attacks by using the sliding window technique and discarding old packets. AH tries to protect all fields of a packet. The ESP provides privacy and protects against malicious modification of a packet. It provides origin authenticity, integrity, and confidentiality of a packet. However, unlike the AH, it does not account for the IP header. IKE protocol is an automated mechanism for secret keys and other protection-related parameters to be exchanged before communication between the end-points commences. In the current specification of IPSec, a security association (SA) is unidirectional and identified by three elements comprising the IP addresses of the packet (or the IP addresses of the outer tunnel if an IPSec tunnel is used), the security protocols (AH or ESP), and a security parameter index (SPI). The SPI is a pointer that references a session key and algorithm used to protect the data being transported.

The main issue in prohibiting MN from using IPSec to establish a secure communication path lies in the fact that IPSec updates its SA on the fly. When there is a change in the entities used for creating this unique identity, the security is violated and the connection will be terminated. This problem arises when packets are sent from a stationary CN to a MN. The SA is only valid at the time of negotiation between CN and MN using the three entities. Once MN enters another network and changes its IP address, the SA will be invalid. Thus, when the destination IP address of a packet changes, a new IKE negotiation must be performed. This implies that when a MN changes its point of Internet attachment, it will have to make new IKE negotiation with its CN. This incurs heavy computation and huge latency which renders mobility of MN impractical.

III. Terminal mobility support protocol

TMSP enables transparency of IP address changes for MN as it moves across networks. MN does not require a permanent IP address and it does not need any home network. It acquires a new CoA whenever it enters a new network. In TMSP, each

MN registers onto the Internet using a SIP URI[1] as a unique identifier. It relies on a SIP redirect server, to provide a URI-to-CoA resolution and a user agent (UA) to handle SIP messages.

The registration of a URI with a CoA and the updating of new CoA for MN are illustrated in section III-B and III-E, respectively. The procedure for resolving the CoA of MN when establishing a data connection and transmitting a packet are detailed in sections III-C and III-D.

Apart from using a UA, TMSP requires an IP-to-IP address mapping module (AMM) to be installed after the IPSec module and an IP header restoring module (HRM) to be installed before the IPSec module in the network layer of the protocol stack in each node. The following sections describe the modules required on MN and its CN to enable TMSP, and the operation of TMSP.

A. TMSP Modules

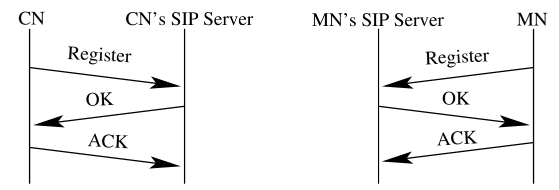
TMSP uses two kernel modules, each taking on a different role to provide IP mobility. They are HRM and AMM. The HRM restores the IP addresses in the IP header of an IPSec packet to the IP addresses used at the instant when the connection was first established, before the IP packet is passed into the IPSec module of the kernel. HRM thus removes the need for establishing an IKE negotiation due to the change in the destination IP address of the data packet due to mobility. The key functions of AMM are to intercept outgoing and incoming IP and IPSec packets at the network layer to perform an IP-to-IP address mapping function. For IPSec packets, it appends/removes a session identifier (SID) onto the packets' AH after/before they are processed by the IPSec function. We will describe the operation of the latter in section III-D.

TMSP keeps states for each data connection in MN and its CN. It maintains three linked state tables. They are the mapping table (M-Table), URI table (U-Table) and SID table (S-Table).

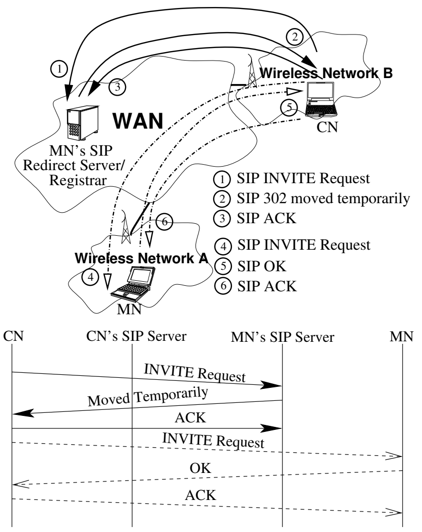
The M-Table maintains an entry for each network port. Each entry contains a reference source IP address (RSA), a reference destination IP address (RDA), a SID of CN, an index to the S-Table, and an index to the U-Table. The U-Table keeps information for each CN. Each entry contains a URI, and a CoA. The S-Table keeps a mapping between each network port and a SID for each network port at MN.

The RSA and RDA refer to the IP addresses used by MN and CN respectively, at the instant when a data connection is established. The CoA of CN refers to the temporary CoA of CN during the lifetime of the data connection. The SID is created for each IPSec data connection. As the network port is encrypted and encapsulated in an IPSec header for each IPSec packet, AMM uses SID as the reference key to locate the network port to apply IP-to-IP address mapping on the IPSec header.

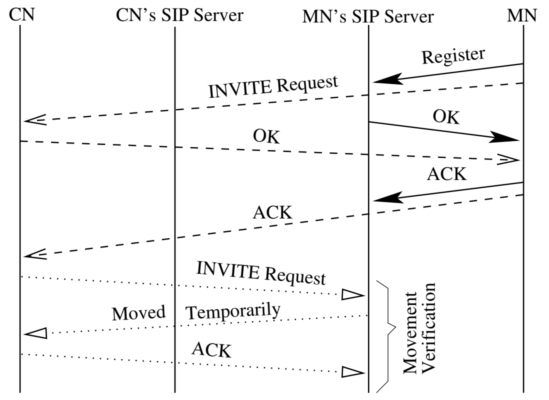
[1]A SIP URI is an email-like address that contains two parts: user and server parts, separated by an "@", e.g. "user@server". The user part contains the user name to identify the user and the server part contains the IP address or domain name of the SIP Server.



(a) Registering the location of MN and CN with their respective SIP-RS.



(b) Establishing a data connection from a CN to a MN.



(c) Updating of the latest acquired CoA of MN

Fig. 1. This figure shows the flow of SIP messages used for (a) making a registration with SIP-RS, (b) establishing a data connection between CN and MN, and (c) updating SIP-RS and CN with the latest acquired CoA of MN after MN enters a foreign network.

B. Boot Strapping

Mobility for a MN begins when its UA completes a SIP registration procedure with MN's SIP redirect server, specified in the server part of the SIP URI as provided by the user. MN and CN use SIP register message to register themselves with their respective SIP-RS. The sequence of message exchanges is shown in fig. 1(a). It is important to highlight that CN and MN do not need to be registered with the same SIP redirect server.

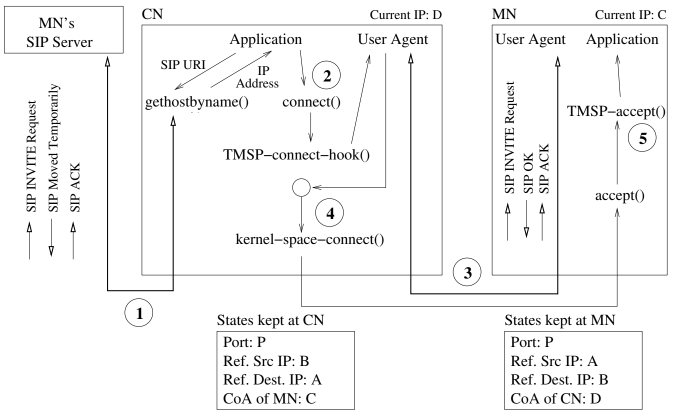


Fig. 2. An example showing the flow for establishing a connection from CN to MN.

C. Connection Establishment

A data connection begins when a CN establishes a connection to a MN by first locating the CoA of MN using the latter's SIP URI. CN's UA uses MN's URI to obtain MN's CoA. It sends a SIP invite request message to the SIP-RS used by the MN as shown in fig. 1(b). In return, the SIP-RS replies with a SIP moved temporarily message containing the CoA of MN. After acquiring MN's CoA, it sends a new SIP invite request message to MN directly to establish a mobility session. Thus, these steps ensure a SIP URI to CoA resolution.

With reference to fig. 2, under the usual SIP operation, for the CN, *gethostbyname()* function is called to map SIP URI to the CoA indicated by label (1), followed by the usual data connection made using the *connect()* function. Thereafter, in the kernel space, the default *kernel-space-connect()* function indicated by label (4) is called to establish connection with the MN. On the MN side, the data connection packet is captured by the *accept* function for TCP for onward conveyance to the application.

TMSP will require the modification of *gethostbyname()* function as well as the incorporation of two new functions, namely, *TMSP-connect-hook()* function and *TMSP-accept* function shown as label (4) and label (5) respectively in fig. 2.

TMSP can automate the process for SIP URI to CoA resolution, for example, by providing a new *gethostname function* or modifying the existing *gethostname function* provided by the operating system, and adding SIP URI as a new family name for network sockets. As shown in fig. 2, an application at CN attempts to get the CoA of MN by calling the *gethostbyname()* function using the MN's SIP URI. We hook on this *gethostbyname()* function to filter for SIP URI requests and call its UA who returns the CoA of MN as indicated by label (1). In the case when SIP URI is added as a family name in network sockets, the *connect()* function will call the UA to resolve the CoA of MN using the given SIP URI. Thus, SIP URI to IP address resolution is completed. After that, application will perform a usual data connection establishment and data transfer by executing the socket function calls without any need to support TMSP operations. In the kernel, TMSP hijacks these functions calls to perform its mobility procedures.

As shown in fig. 2, a usual data connection is made using the *connect()* function as indicated by label (2). In the case when SIP URI is added as a family name in network sockets, the *connect()* function will call the UA to resolve the CoA of MN using the given SIP URI. Thus, SIP URI to IP address resolution is completed.

In the kernel space, our *TMSP-connect-hook()* function is called before the default *kernel-space-connect()* function to generate a unique SID for this data connection if IPSec is the requested protocol, and to call its UA to perform a SIP invite request procedure to MN as indicated by label (3). The UA sends a new SIP invite request message directly to the UA of MN, who in return, replies with a SIP okay message. For a data connection that uses IPSec, the SIP invite request and okay messages will contain the CN's SID and MN's SID, respectively. To enable this exchange of SID, we add a "tag" in the SIP invite request and okay messages. Thereafter, the *kernel-space-connect()* function is called and this data connection establishment continues without further alteration as indicated by label (4). During the SIP invite request procedure, the UA of both CN and MN record the SIP URI, the CoA and SID of each other in their state tables.

The packet used to establish this data connection is captured at MN's *TMSP-accept* function, as indicated by label (5), after the *accept* function for TCP, to register the network port number used for the connection and update the state tables. On receiving the packet that indicates the acknowledgment of the data connection, our *TMSP-connect-hook* function at CN, updates the state tables based on the network port number used for the connection. Hence, once data connection is established, the state tables are updated and ready to perform an IP-to-IP address mapping for subsequent packets.

D. Pathway of IP and IPSec Packet

To illustrate TMSP's support for IP mobility, we show the path of an IP packet or IPSec packet from the moment it enters the HRM in CN (the sender) to the point it is passed to the higher layer of the protocol stack after the AMM in MN (the recipient). Figs. 3 and 4 show examples of an IP packet and an IPSec packet generated by a connection between CN and MN when both are attached to the Internet via a wireless access point (AP). This connection is previously established when CN and MN are holding CoA, *A* and *B*, respectively. At the instant when this packet is generated, CN has acquired a new CoA, *C*, from AP_M, while MN has acquired a new CoA, *D*, from AP_N. Both MN and CN have also informed each other about their new CoA using a SIP invite request procedure discussed in section III-E. Thus, for this connection, the *RSA* is *A*, *RDA* is *B*, and MN's CoA is *D* in CN's state table; and the *RSA* is *B*, *RDA* is *A*, and CN's CoA is *C* in MN's state table. As shown in the figures, the application at CN is unaware that MN has changed its CoA from *B* to *D*, and it continues to use the *RDA*, *B*, as parameters for the network layer to generate IP packets towards MN. The network layer generates the IP packet using *B* as the destination IP address and the current CoA of CN as the source IP address. This IP packet bypasses

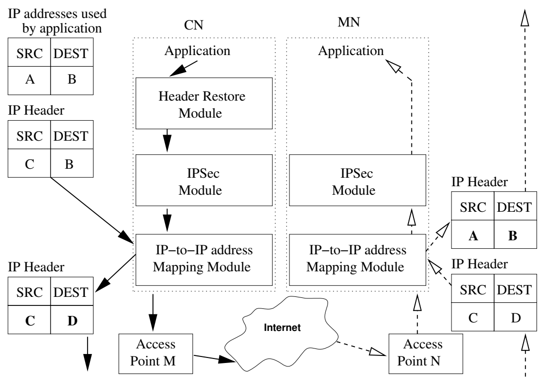


Fig. 3. Pathway of an IP packet from a CN to a MN.

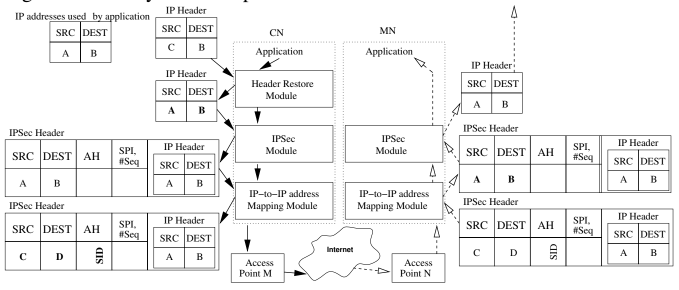


Fig. 4. Pathway of an IPSec packet from a CN to a MN.

the HRM and the IPSec module as both modules only process IPSec packets.

For IPSec packet, the HRM will replace the source IP address in the IP packet's header with the RSA. This restores the IP header that is used to generate security checks for IPSec. Without this IP header restoration, the IP packet will fail the IPSec module's security check. This method allows IPSec module to use the share key that is established at the instant when the connection is established. This module also grabs the network port number used by the data connection and temporary stores it for the AMM, which operates after the IPSec module in the protocol stack. The temporarily storage of the network port number is necessary as the IPSec module will encrypt the packet as a payload and encapsulate it with an IPSec header (see fig. 4).

The AMM replaces [2] the destination IP address of the IP or IPSec packet by the CoA of the MN based on its entry in the M-Table, referenced by the network port number or the temporarily stored network port number, respectively. It also replaces the source IP address to CN's CoA. This is necessary to prevent network ingress filtering [9]. Finally, for IPSec packet, it pastes the MN's SID into the reserved bits of the AH (see fig. 4).

This packet is then directly routed to the MN. At the network layer of MN, the AMM replaces the source and destination IP addresses of the IP or IPSec packet by the *DSA* and *RSA*, respectively, as stored in its entry in its M-Table, referenced by port number for IP packet or SID for IPSec packet. After that, this packet is sent to the higher layer of the protocol stack. The IPSec packet is decrypted at the IPSec module before sending the packet to the higher layer of the protocol stack. There is no further IP-to-IP address mapping on the decrypted packet as the HRM at CN has restored the IP header of this packet to the state when the connection is first established.

[2]The checksum for the network layer and transport layer of a packet must be recomputed if changes to the IP header is made. This is to prevent dropping of malformed packets before they are sent to the Internet.

IP address transparency is achieved using the IP-to-IP address mapping method. This method ensures that a packet is restored to its original IP address used by the application at the instant when the connection is established. Thus, applications do not need to be aware of the CoA changes on the MN.

E. Updating newly acquired CoA

When MN enters the coverage of another foreign network, it will acquire a new IP address from the network and start to use it as its CoA. Fig. 1(c) shows the sequence of SIP messages used by MN to inform its SIP-RS and re-invites its CNs after it moves to another network. MN uses the SIP registration procedure to inform the change of location to its SIP-RS. It re-invites its CN using the SIP invite request message to report the newly acquired CoA. Upon receiving this re-invitation from MN, CN will update the MN's CoA in its U-Table. The latter is used by the AMM to deliver packets to the current IP address of MN.

In TMSP, SIP messages used can be protected by IPSec. To ensure an even more acceptable form for CoA update, CN's UA can start a simple movement verification procedure (see fig. 1(c)) by matching new MN's CoA with the one acquired from MN's SIP-RS after the re-invitation. Alternatively, SIP-RS can take the role to inform all CNs on behalf of MN to re-assure the CNs that the re-invitation is genuine after MN has informed it of the location change. This method, however, requires SIP-RS to perform a new function which adds processing load, which requires the use of a modified set of SIP register message to include all the CoA of CNs.

There is a possible scenario where the operation of TMSP may be disrupted. When MN and its CN move to a new network concurrently, SIP invite request messages sent, both by MN and CN, may not be received, i.e., MN and CN do not receive the new CoA of each other. When this happens, the UA at MN can still discover the CoA of CN via CN's SIP-RS using CN's URI, and send another SIP invite request message to CN. Likewise, CN can perform the same procedure to rediscover MN. This recovery process can be triggered by the timeout set for SIP invite request procedure. MN's UA can also start resolving for CN's CoA via CN's SIP-RS after a fraction of the timeout for SIP invite request procedure. This method will certainly shorten the recovery time.

IV. Comparison

To analyze the performances of TMSP compared to the MIPv6 Basic mode and an application-layer mobility protocol using SIP-MIPv6, we classify the protocols into their signaling cost and transmission cost. Signaling cost can be broken down into movement cost and session maintenance cost. When a MN moves, a series of message exchanges begins to keep the MN's movement up-to-date. Movement cost, $C_{movement}$, is defined as the total number of hops that these messages traversed. Session maintenance cost, $C_{maintain}$, records the total number of hops that messages traversed to maintain existing sessions between CNs and MN. Lastly, transmission cost, C_{tx}, counts the number of hops that a message needs to traverse from CN to MN.

We compare TMSP with MIPv6 and an integration of SIP with MIPv6 (SIP-MIPv6) using the metrics defined above. This integration is proposed by E. Wedkund and J. Schulzrinne [10] for mobility support of both TCP and UDP using SIP. For MIPv6, we assume that co-located CoA is used and we use D_{A-B} to denote the number of hops between network entity A and B.

When MN moves into a foreign network, it acquires a new IP address via DHCP. In TMSP, it starts a registration procedure using a message pair (Register and Okay messages) with its SIP-RS. While in MIPv6, it starts a binding update procedure using a message pair (binding update and acknowledgment messages) with its HA. In SIP-MIPv6, it triggers both the registration and the binding update procedures.

The $C_{movement}$ for TMSP, MIPv6 and SIP-MIPv6 are shown in (1), (2) and (3), respectively.

$$\begin{aligned} C^{TMSP}_{movement} &= 4D_{MN-DHCP} + 2D_{MN-RS} && (1) \\ C^{MIPv6}_{movement} &= 4D_{MN-DHCP} + 2D_{MN-HA} && (2) \\ C^{SIP-MIPv6}_{movement} &= 4D_{MN-DHCP} + 2D_{MN-RS} && (3) \\ & \quad +2D_{MN-HA} \end{aligned}$$

When MN moves into a new foreign network, it informs its CNs of its new CoA to continue existing sessions. In TMSP and SIP-MIPv6, MN informs its CNs using the SIP invite request messages that consists of a message pair (SIP invite request and okay messages). In MIPv6, MN does not inform its CNs of its new IP address. Thus, it does not incur any signaling cost. The $C_{maintain}$ for TMSP, MIPv6 and SIP-MIPv6 are shown in eqn. 4, 5 and 6, respectively where η is the number of CN.

$$\begin{aligned} C^{TMSP}_{maintain} &= 2\eta D_{MN-CN} && (4) \\ C^{MIPv6}_{maintain} &= 0 && (5) \\ C^{SIP-MIPv6}_{maintain} &= 2\eta D_{MN-CN} && (6) \end{aligned}$$

When TMSP is used, a message transmitted from CN to MN follows the conventional IP routing for both TCP and UDP packets. When MIPv6 is used, a message transmitted from CN to MN is first sent to the MN's HA where the message is tunneled to MN. Likewise, a message from MN to CN is reverse tunneled back to the MN's HA before it is sent to the CN. When SIP-MIPv6 is used, the message route path is similar to TMSP for UDP packet and is similar to MIPv6 for TCP packet. The C_{tx} for TMSP, MIPv6 and SIP-MIPv6 are shown in eqn. 7, 8, 9 and 10.

$$\begin{aligned} C^{TMSP}_{tx} &= D_{MN-CN} && (7) \\ C^{MIPv6}_{tx} &= D_{MN-HA} + D_{CN-HA} && (8) \\ C^{SIP-MIPv6}_{tx,UDP} &= D_{MN-CN} && (9) \\ C^{SIP-MIPv6}_{tx,TCP} &= D_{MN-HA} + D_{CN-HA} && (10) \end{aligned}$$

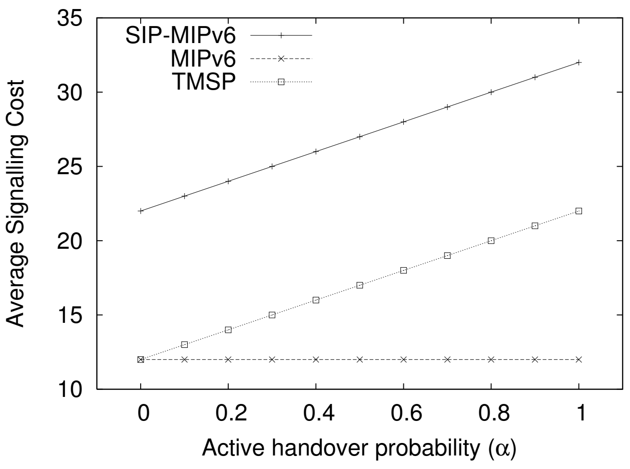


Fig. 5. This figure shows the average signaling cost when network nodes use TMSP, MIPv6 and SIP-MIPv6.

For simplicity of comparison, we assume that DHCP server is always one hop away from MN and there are five hops between two network entities. We evaluate the performances of TMSP over MIPv6 and SIP-MIPv6 using the cost functions defined.

A. Average signaling cost

From the definition of $C_{movement}$ and $C_{maintain}$, we can derive that the cost of a handoff[3] , $C_{signalling} = C_{movement} + C_{maintain}$. Define α as the active handoff probability that a MN visits a foreign network with active connections. Thus the idle handoff probability will be $(1-\alpha)$. The average signaling cost, $\widehat{C_{signalling}} = (1-\alpha)\ C_{movement} + \alpha\ C_{maintain}$. Fig. 5 plots the average signaling cost. Since MIPv6 only performs binding update to the HA, the $\widehat{C_{signalling}}$ is a constant and is the lowest as compared to TMSP and SIP-MIPv6. Both TMSP and SIP-MIPv6 perform binding update to CN if there are existing connections in addition to HA. Therefore, $\widehat{C_{signalling}}$ of TMSP and SIP-MIPv6 increase correspondingly with an increase in α due to the additional maintenance cost, $C_{maintain}$, required to update CN with the latest CoA.

B. Data transmission gain

To better reflect the relative gain in transmission cost of TMSP over the other protocols, transmission gain is used in the evaluation. The transmission gain is defined as the reduction in hop counts of TMSP when compared to another protocol, and it is expressed in percentage. The hop count is measured from the point a packet is sent from MN to CN after MN has moved into a new foreign network (i.e., it includes the cost of one movement cost function). The transmission gain for sending varying number of packets from MN to CN, which includes one handover process as defined in section IV above, is plotted in fig. 6.

[3]The cost of a handoff in this context refers to the duration from the point a host enters a new network until it has successfully attained a new IP address.

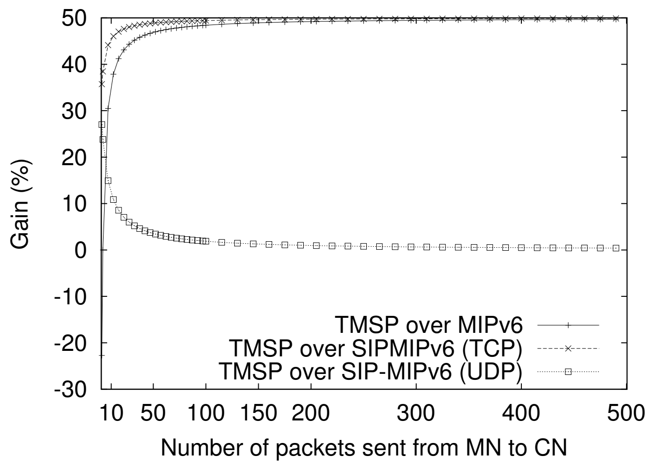


Fig. 6. This figure shows the gain in data transmission when network nodes use TMSP as compared to MIPv6 and SIP-MIPv6, including the overhead cost of a MN moving into a new foreign network. This plot assumes that MN only has one CN.

From fig. 6, TMSP only begins to have a transmission gain over MIPv6 when MN sends more than two packets to CN. This corresponds to the characteristics of the protocols as TMSP incurs an additional maintenance cost, $C_{maintian}$, to inform CN about MN's movement into a new foreign network. In TMSP, MN sends packets directly to CN using the conventional IP routing, thus it has a lower C_{tx} as compared to MIPv6 where which packets sent between CN and MN need to pass through MN's HA. TMSP has a transmission gain over SIP-MIPv6 for both TCP and UDP packets. For TCP packets, TMSP has an increasing transmission gain function over SIP-MIPv6 as the latter incurs higher cost in $C_{movement}$ and C_{tx}. For UDP packets, TMSP has a decreasing transmission gain function (tending to zero) over SIP-MIPv6 as the latter only incurs a higher cost in $C_{movement}$ while it shares the same C_{tx}.

C. Traffic overhead

We evaluate the advantage of TMSP over MIPv6 basic mode quantitatively by comparing the network overhead generated by a CBR traffic source. The CBR source generates a total of $T = R \times (8L) \times t$ bits for a duration of ts, where R is the rate (packets/second) of the traffic source and L is the size of a IP packet.

For MIPv6, the amount of traffic generated onto the network from CN to MN is given by,

$$\begin{aligned} T^{MIPv6} &= 8RtLH^{MIPv6}_{CN-HA} \\ &\quad +8Rt(L + L^{IP-in-IP})H^{MIPv6}_{HA-MN}, \end{aligned} \quad (11)$$

where $L^{IP-in-IP}$ is the size of the outer IP header used to encapsulate the packet, H^{MIPv6}_{CN-HA} is the number of hops from CN to the MN's HA and H^{MIPv6}_{HA-MN} is the number of hops between MN and its HA. In MIPv6, the HA of MN is a fixed redirection point for exchange of every IP packet between CN and MN. For TMSP, the amount of traffic generated onto the

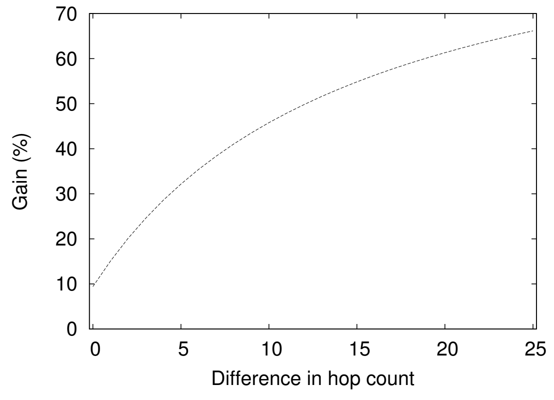


Fig. 7. This figure shows the gain in terms of traffic generated when TMSP is used over MIPv6. The x-axis indicates the difference in hops between the triangular route and the direct route CN-MN.

network from CN to MN is given by,

$$T^{TMSP} = 8RtLH^{TMSP}, \quad (12)$$

where H^{TMSP} is the number of hops between CN to MN.

In this comparison, we assume that IPv4 is used. Thus, the $L^{IP-in-IP} = 20$ bytes. We use a 2-min VoIP session that generates a flow of 89 180-byte packets per second, yielding a rate of $128kbps$. Fig. 7 shows the gain ($\frac{T^{MIPv6} - T^{TMSP}}{T^{MIPv6}}$) from (11) and (12)) for using TMSP over MIPv6 in terms of the total traffic generated using the VoIP session, where CN to MN is 15 hops away[4] in the case of TMSP and a variation of 15 to 25 hops away in the case of MIPv6 such that the CN is always 1 hop away from HA of MN. The parameters H^{TMSP} is set to 5, H^{MIPv6}_{CN-HA} is set to 1, and H^{MIPv6}_{HA-MN} is a variable from 14 to 24. From the figure, the first plot point shows that there is a gain using TMSP even when the number of hops are the same (i.e., $H^{TMSP} = H^{MIPv6}_{CN-HA} + H^{MIPv6}_{HA-MN}$). This is because every IP datagrams is encapsulated with an outer IP header at the HA before it is redirected to MN. This gain increases further as the number of hops between MN and its HA increases from 14 to 24. To give a further comparison, we plotted the gain (see fig. 8) in the same scenario as above but with a maximum triangular route of 30 hops distributed in different proportion between H^{MIPv6}_{CN-HA} and H^{MIPv6}_{HA-MN}. The x-axis denotes the distribution of the number of hops between H^{MIPv6}_{CN-HA} over H^{MIPv6}_{HA-MN}. Each plot indicates the gain for using TMSP when the direct route between CN and MN has $10, 15, 20$ and 25 hops. Fig 8 shows that the gain increases as the number of hops due to triangular routing increases. Gain increases are also registered with increasing number of hops between MN and its HA. In fact, this gain also increases with the duration of the VoIP session and increasing number of traffic sessions. Thus, the use of direct route offered by TMSP eradicates the inefficiency caused by triangular routing and the additional use of outer IP header to encapsulate every IP packet.

Before IP packet can be routed directly between CN and

[4]Estimated based on the measurement of average hop counts of the Internet reported by A. Fei et. al. [11].

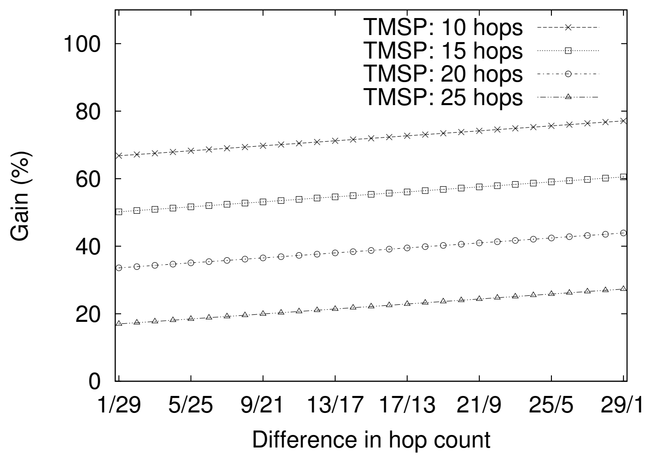


Fig. 8. This figure shows the gain of using TMSP when the number of hops for the direct route is fixed at 10, 15, 20, 25 and the number of hops for the triangular route is fixed at 30 but distributed in proportion ($H^{MIPv6}_{CN-HA}/H^{MIPv6}_{HA-MN}$).

MN, the route optimized mode in MIPv6 still relies on HA's redirection of IP datagrams before MN completes a binding update process in which it informs CN of its CoA. So, the initial N IP datagrams will suffer from triangular routing. A Home Address option carried by the Destination Option extension header is used in a IP datagram sent by a MN which is away from HA, to inform the CN of the MN's permanent IP address. A Type 2 Routing Header is used by CN to allow IP datagram to be routed directly to the CoA of MN. This header also contains the permanent IP address of MN who will in turn replace the destination address of the IP datagram with its permanent IP address. Both headers require an overhead of 48 bits to encapsulate the permanent IP address of the MN which is 128 bits. Thus, without considering the traffic incurred by the initial N IP datagrams and assuming that the number of hops between CN and MN are the same, each IP datagram is 44 bytes larger in MIPv6 as compared to TMSP.

V. Applicability of TMSP

The requirement of upgrading operating system module to support TMSP and running a SIP UA on end hosts may appear to be unattractive. However, this is not different from other mobility schemes. For direct communication (e.g., VoIP, file exchange) between two clients (mobile or non-mobile), TMSP provides a perfect solution for seamless communications. TMSP is the enabler for IP mobility management when host roams between homogeneous and heterogeneous wireless networks.

Another possible critique of TMSP is that it requires changes to existing servers to provide mobility. However, this is not different from several other mobility schemes that provide route optimized mobility solutions as discussed. A TMSP-enabled proxy can be designed and implemented to enable legacy servers to join TMSP-enabled network without upgrading. Furthermore, we also support the argument in [12] that not all applications require network-layer mobility, especially those characterized as short transactions where an application level retry of the transaction is easy to perform (e.g., reloading a web page). TMSP works for point to point communication software like VoIP and video conferencing. These are key applications of IP convergence in communications.

TMSP-enabled hosts continues to operate with the current Internet. It does not affect the operation of legacy applications running on a non-TMSP-enabled server, like web browser, or hosts. This enables a smooth transition and motivation for manufacturers to include TMSP as a pre-loaded module in their products.

One of the biggest limitation of TMSP is that there can exist a scenario that the IP addresses of both mobile hosts are not updated correctly when both mobile hosts move and change their IP address simultaneously. This scenario will affect the handover delay as the IP addresses of corresponding mobile hosts are being recovered via the movement verification process as demonstrated in fig. 1(c).

Finally, due to the reuse of IP addresses, there exists very slight possibility of IP address mapping confusion for connectionless service hosted by UDP server that can receive packets from many clients on a single port bounded to the server socket. This server uses the *recvfrom()* function to track the client's IP address and port for sending back a reply. A mobile host (MH1) holding on IP address B, and has previously established and created a mapping of $(A \rightarrow B, P)$ on this server when it was holding IP address A will cause an IP-to-IP address mapping confusion with another mobile host (MH2) that attempts to connect to this UDP server after it obtains IP address A using the same port number, P. In this scenario, MH2 will not be able to receive replies from the server as all replies are routed to MH1 based on the IP-to-IP address mapping created by MH1 on the server. Such server usually does not provide any long-live connections. Examples of such server include echo, charge and time. In such a scenario, TMSP can be configured to void IP-to-IP address mapping for selected ports or reduce the chances of such confusion by setting a fast time up for UDP ports in the M-Table.

VI. Implementation and Results

We implemented TMSP on Linux kernel 2.6. We built a kernel module to perform the HRM, AMM and a null character device driver to store the state tables so that state information can be accessed from both the kernel and user spaces.

The HRM hooks before the IPSec module in the kernel intercept outgoing IP packets. The AMM hooks after the IPSec module in the kernel to intercept outgoing IP packets, and before the IPSec module in the kernel to intercept incoming IP packets as shown in fig. 3.

Briefly, for outgoing IP packets, the HRM decodes each IPSec packet for the network port number used and temporarily stores it in the socket buffer. The AMM uses network port number to perform IP-to-IP address mapping on the IP header as described in section III-D using the state tables, and stuffs CN's SID (obtained from the M-Table) into the reserved bits in the AH if the packet is an IPSec packet.

For incoming IP packets, the AMM decodes the network port number and performs an IP-to-IP address mapping on the packet header. As for IPSec packet, it uses the SID in the AH to obtain the network port number before it performs an IP-to-IP address mapping on the packet header.

In addition, the routing cache must be cleared when the MN changes its network attachment point, i.e. whenever MN has acquired a new CoA. This is necessary as the Kernel caches routes for TCP connections which can forward TCP packet to the previous gateway that may not exist in the newly joined network.

We also hooked on several kernel primitive functions for communication like *connect*, *listen*, *close*, *sendmsg* and *recvmsg* to detect for new connections and closed connections to update the state tables. Finally, the UA is implemented as a daemon process that listens at a fixed port (15000) for all incoming SIP messages from any CN and SIP-RS.

The following sections present the experimental results for TMSP. Section VI-A reports the effect of installing TMSP on a MN. Section VI-B records the experiments carried out to study and show the performance of TMSP in supporting IP mobility.

A. Effect of TMSP on a MN

1) Effect of TMSP on the throughput of WLAN: We conduct an experiment to show that our TMSP implementation does not affect the throughput of a WiFi interface. Experiments are run to compare the saturation throughput of a MN using a IEEE 802.11g WLAN interface. MN floods the air channel using a CBR traffic of $30Mbps$. Fig. 9 shows the average throughput achieved by a MN when it floods its wireless interface with TCP and UDP packets of different sizes for both IP and IPSec networks. The reported throughput are approximately equal regardless of whether TMSP is installed on MN. Thus, this shows that TMSP does not affect the achievable throughput of the WLAN in this scenario. The IP-to-IP address mapping has negligible impact on the WLAN throughput. In addition, we also verified this measurement quantitatively using an analytical model [13] by G. Bianchi as shown in the figures. Hence, TMSP shows that it does not compromise the achievable bandwidth of WLAN.

2) Performance of the state tables: We tested the performance of the null-character device driver that stores the state tables for TMSP by performing the following functions: add, access, update and delete. We generated new entries randomly to be added to the state table. An experiment is conducted by measuring the time taken to add a random entry to the state tables when the number of entries in the M-Table vary from 1 to 600. State tables take an average of $0.335ms$ to add an entry from the UA (i.e., from the user space). Likewise, another experiment is carried out to randomly remove an entry. The average time taken to remove an entry is $5.24\mu s$ in kernel space. When a MN detects a change in network attachment point, it will access all entries in the U-Table and send SIP invite requests to all its CN. This access copies the entries from the kernel space to the user space. The time taken to retrieve all entries in the M-Table is shown in table I. Upon receiving

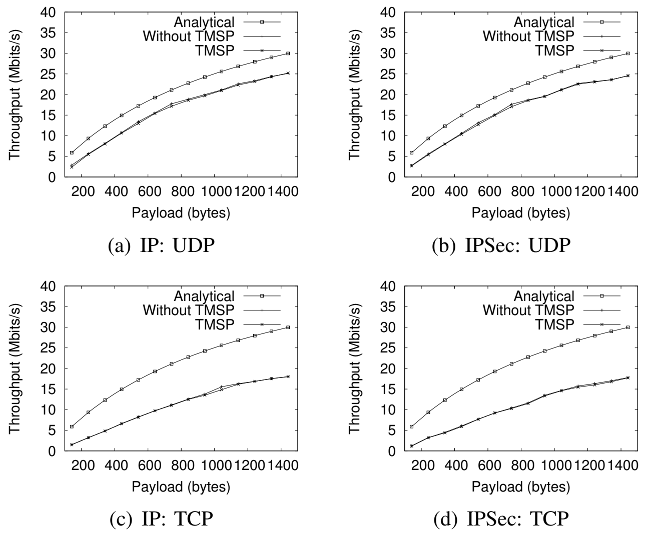


Fig. 9. Average throughput (Mbps) recorded at the wireless channel when the flooding CBR source uses different datagram sizes (bytes)

TABLE I
OPERATION COST FOR THE STATE TABLES (ms)

Entries	1	100	250	500	750	1000
Accessing all entries	8	73	185	341	506	682
Updating an entry	6	9	12	18	24	29

a SIP invite request message, a CN needs to update the U-Table. An experiment is conducted whereby the updating entry is generated randomly with equal probability in the U-Table. The time taken for this update operation by CN's UA is also shown in table I. In our implementation, an average delay of $0.335ms$ contributes to the initial setup time required by TMSP to react to the incoming and outgoing IP packets for the new connection. When MN moves, CN requires an adaptation delay to respond to the change of IP address for MN after receiving a SIP invite request message. Therefore, the average delay to update an entry in the state tables is the average adaptation delay for TMSP; while the time required to access all entries in the state tables is the worst-case adaptation delay for TMSP. In our implementation, it can take as long as $682ms$ using sequential search to find an entry in the state tables at the worst scenario. This long delay can be greatly reduced by optimizing data structure and search algorithms.

3) Performances of the IP-to-IP address mapping: This section reports the operational performance of the AMM at the network layer of the protocol stack. The following experiment shows the delay incurred in processing IP packets. The experiment measures the duration before a packet enters the AMM to the time this packet leaves the network layer in the protocol stack in a MN. This is carried out by adding two timing checks in the kernel, one at the beginning of the IP input routine and the other at the end of the IP output routine. Fig. 10 compares the average packet processing time of the network layer in the protocol stack to process a packet under the two scenarios: 1)

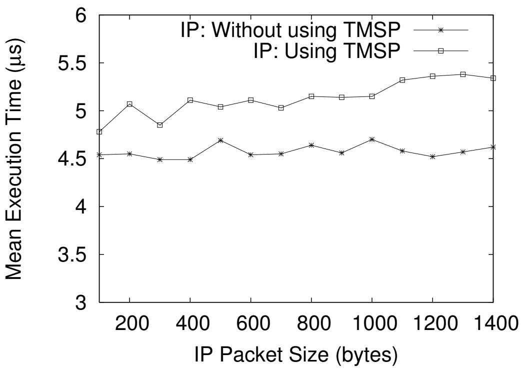


Fig. 10. This figure shows the packet processing time for a kernel when TMSP is not loaded, and when TMSP is loaded. All packets require IP-to-IP address mapping.

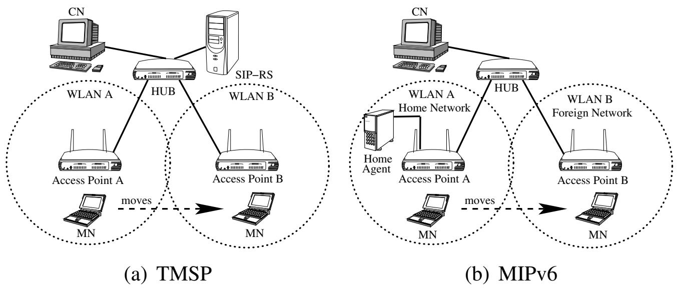


Fig. 11. Test-bed used for collecting experimental results for TMSP and MIPv6

MN does not load TMSP and, 2) MN loads TMSP and all packets require IP-to-IP address mapping. The average packet processing time for a kernel without TMSP loaded is $4.57\mu s$. With TMSP loaded, the time for processing packets is $5.13\mu s$. which increases the processing time by about 12% or $0.56\mu s$. The transmission delay for the minimum frame size (64 bytes) of ethernet on a $100Mbps$ LAN is approximately $5.12\mu s$ and that for the minimum frame size (512 bytes) of gigabit ethernet on a $1Gbps$ LAN is approximately $4.10\mu s$. The additional delay of $0.56\mu s$ introduced by TMSP is therefore negligible as it is much smaller than the transmission delay of the minimum frame size of ethernet. Thus, TMSP will not affect the transmission rate of a host. Additionally, this experiment also shows that the packet processing delay does not depend on the packet size.

B. Mobility

We have successfully tested TMSP with video streaming and video conferencing applications. We study the mobility of TMSP and MIPv6 using the test-beds depicted in fig. 11(a). It comprises two APs, a SIP-RS, a CN and a MN. Another test-bed, depicted in fig. 11(b), is used to test MIPv6. The MN is a $1.86GHz$ centrino processor laptop, the CN and the HA are 2.4GHz desktop computers, and the APs are linksys WRT54G running openwrt at 54Mbps.

1) Effect of crossing networks with an active connection: This section details an experiment setup to illustrate that TMSP is able to resume an active connection when a MN travels from its current network to a neighbouring network. In this experiment, a UDP session using a traffic source from MN to CN is maintained while MN moves from Wireless LAN A to Wireless LAN B. Measurements taken at MN are plotted in fig. 12. Fig. 12(a) to 12(b) plot the packet sequence numbers received at MN against time for a UDP connection. Each plot in fig. 12 shows that TMSP allows MN to continue receiving packets from CN after it changes its point of attachment from Wireless LAN A to Wireless LAN B at t=$480s$. The same results apply for TMSP when IPSec is used. TMSP recorded an average handover of $2s$, while MIPv6 recorded an average handover of $3.35s$.

In TMSP, the handover latency comprises three timings. They are movement detection time, new IP address acquisition time and SIP re-invitation processing time. The first two timings depend on whether the handover is performed using a single wireless interface or multiple wireless interfaces. The time required to detect movement into another network in MN and to obtain an IPv4 configuration can be significant in the total handover latency when a MN moves between points of attachment. In these experiments, we polled for any new association with a wireless LAN for the wireless interface of MN at an interval of $0.2s$. We recorded a delay of $1.7s$ to associate and acquire an IP address for the wireless interface of MN with a wireless access point running DHCP, and an average of $96ms$ to complete a SIP invite request procedure.

There is a standard proposed to introduce a set of steps known as DNAv4 [14], in order to decrease the handover latency in moving between points of attachment. There is also a standard protocol proposed to improve the delay in new IP addresses acquisition for DHCP using a rapid commit option [15] whereby the usual 4-message exchange is reduced to a 2-message exchange.

When there is overlapping wireless coverage, a make-before-break technique will be able to provide a seamless handover and minimize packet lost. We illustrated this technique using two WLAN interface cards. Fig. 12(d) shows that there is no packet lost when MN pre-establishes a connection with WLAN B using another WLAN interface card. In this experiment, the association with WLAN A only breaks after MN receives packets from WLAN B.

A make-before-break handover using a single wireless LAN interface can be possible using Multinet [16]. It multiplexes a single network interface for connections to more than one infrastructure WLANs. This mechanism allows a MN to connect to two WLANs and performs early binding update before fully switching to the new network attachment point.

There are network-assisted approaches that require APs to participate in the handover process for MN to minimize packet losses in handover. Fast handovers for MIPv6 [17] introduces a protocol to improve handover latency due to MIPv6 procedures, and HMIPv6 reduces the signaling delay incurred by informing HA and/or CN of any change in IP address of MN using the hierarchical nature of IP addressing within the visited network. Using a *forced handover*, we further show the seamless connectivity that TMSP provides by switching a MN between wireless LAN A and B at every $120s$-interval. We performed a forced handover by manually

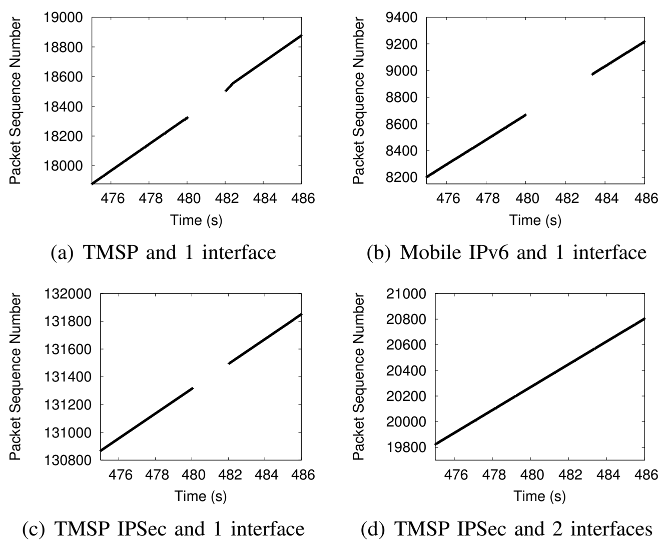


Fig. 12. This figure shows the number of packet received at MN for a UDP connection from CN to MN when MN switches between WLAN A and WLAN B.

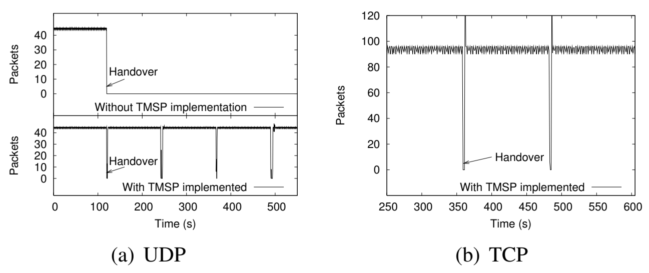


Fig. 13. These figures show the number of packets received at MN at every 100ms interval when it switches between WLAN A and WLAN B from the start of the experiment for (a) UDP connection and (b) TCP connection. For the top figure in (a), MN does not implement TMSP. For the bottom figure in (a) and figure (b), MN implements TMSP.

changing the "ESSID" of the wireless interface, re-assigning IP address and altering the default route of the iptables on the MN.

In these experiments, a transport session from CN to MN is maintained when MN is on-the-move. Fig. 13(a) and 13(b) plots the throughput of the transport session when TMSP is not implemented, and when TMSP is used to maintain a UDP session and a TCP session. Fig. 13(a) plots the number of packets received at every 100ms interval by the UDP session at MN. The bottom figure shows that MN with TMSP installed only encounters a very short duration (in hundreds of milliseconds) of data loss after it moves to another WLAN. The small breakage time is due to the loss in connection with the previous WLAN. The connection resumes right after MN has successfully configured a new IP address in WLAN B and informed CN of this new IP address. On the other hand, without TMSP, the UDP connection breaks right after MN moves to WLAN B at time $120s$ as shown in the top figure in fig. 13(a). TMSP also work on TCP connection as shown in fig. 13(b). Similar to the previous experiment, the TCP connection suffers from a small breakage during the handover process of MN, and it continues with breaking the connection.

VII. Related Work and Discussion

A. Comparison with existing solutions

1) Mobile IP: IP mobility support is first introduced over a decade ago and is subsequently updated for IPv6 and IPv6 networks. It is initially designed based on the principle that fixed Internet hosts and applications are to remain unmodified. Although mobile IP provides a solution for terminal mobility, it has several limitations given that it is conceived quite some time ago without the benefit of recent technological process. These limitations include triangle routing and encapsulation overhead. It also requires a permanent IP address for each MN. Since the HA serves as a point of packet redirection for all its MNs, the network links connecting the HA to the network can be easily overloaded as the packets destined to the same sub-network field converge to the same HA. Mobile IP uses IP-in-IP encapsulation [6] on its packets. This increases the size of each packet.

TMSP does not have this overhead as it does not require tunneling. It also does not require any permanent IP address for each MN. The current IP addresses used by MN and its CN when a connection is first established are used as reference IP addresses for the connection. TMSP uses an IP-to-IP address mapping function in the network layer of the protocol stack to provide IP address transparency to higher layers. In TMSP, all packets from any node are routed directly to MN. Thus, inefficiency caused by triangular routing and packet redirection are nonexistent. However, TMSP requires fixed Internet hosts to update its protocol stacks to support mobility.

The mobile IP's route optimized mode avoids triangular routing and packet direction once MN establishes a binding update with CN. However, it introduces IPv6 extension headers on each packet. This adds an overhead to deliver each packet between MN and CN. TMSP does not introduce any IPv6 extension header and there is no need to rely on a HA to redirect packets.

2) Application-layer mobility for SIP application: Merging the proposed application-layer mobility using SIP [10] with mobile IP to support IP mobility for TCP connections caused double signaling costs. When a MN moves to a new foreign network, it sends a mobile IP binding update to its HA and also a register message to its SIP-RS. In addition, if the route optimized mode is used, It will send binding update and SIP invite request messages to all its CNs.

TMSP works for both UDP and TCP connections. A MN only requires to update its new IP address with its SIP-RS and a SIP invite request to all its CNs.

3) Transport-layer mobility Solution: Instead of solving host mobility using network and application layer, Snoeren and Balakrishnan have presented a solution using the transport layer [12]. Their objective is to retain existing TCP connections using secure and efficient connection migration, enabling established connections to seamlessly negotiate a change in end-point IP addresses without the need for a third party. This solution uses dynamic DNS and a set of novel migrate options, included in SYN segments of TCP that identifies a STN packet as part of a previously established connection, rather than a request for a new connection. This migrate option contains a token that identifies a previously established connection on the same destination identified using [destination address, port] pair, which is negotiated during initial connection establishment. A connection can be uniquely identified by using the triple [source address, source port, token] on each host.

This identification enables a mobile host to restart a previously established TCP connection from a new IP address by sending a special migrate SYN packet that contains the token.

This solution does not handle applications that uses UDP as transport. Applications are required to retry failed transactions whenever hosts change their IP address due to mobility. As discussed by the authors, using this solution will require a mobile host to dismount and re-mount Network File Systems (NFS) upon reconnection when NFS operates over UDP as NFS uses IP address as mounting points.

4) MobileNAT: MobileNAT [18] supports efficient micro and macro mobility of devices across private and public heterogeneous address spaces. This technique uses two IP addresses for each host. A fixed unique virtual IP address for host identification and a dynamic, unique actual IP address for routing within the domain. Address translation is performed both at the client in sub-layer between the TCP/IP stack and the network interface driver.

MobileNAT maintains mobility information of hosts using a set of servers. It suggests modification on DHCP server and relays to issue two per-host IP addresses and to signal mobility events. It also introduces an anchor node (AN), a address translator, and a mobility manager (MM) which is a new signalling entitiy. AN can be implemented in traditional edge router with NAT support or as a separate NAT device connected to a traditional router. MM controls several ANs in one or more domains and is used to signal mobility events to ANs. MM talks to ANs using the middlebox communication protocol (MIDCOM) framework [19] over secure communciations. When the IP address pair corresponding to a MN changes due to mobility, the DHCP server or the MN conveys the mobility event to MM which in turns uses its MIDCOM functions to signal the changes in the mapping rules to AN.

TMSP uses an actual IP address which is unique. At the point when a connection is established, this address acts as the virtual IP address or host identification at the moment for this connection. This removes the need to assign permanent virtual address or identifier to each host. A host can still be located based on the host SIP URI which has a binding to its unique actual IP address at its SIP server. Leveraging on the pervasivness of SIP and also Session Border Controller (SBC) for NAT and firewall, TMSP eliminates the requirement to install new entity to support mobility manangement.

5) Host Identity Protocol: The host identity protocol (HIP) [20] fills an important gap between the IP and DNS namespaces. This architecture decouples the transport layer (TCP, UDP, etc.) from the internetworking layer (IPv4 and IPv6) by using public/private key pairs, instead of IP addresses, as host identities.

Hosts include an overlying protocol sub-layer in its protocol stack. This sub-layer (e.g., transport layer sockets and ESP Security Associations) is bounded to HIP's representation of host identities, and the IP addresses are only used for packet forwarding.

Thus, the upper-layer protocols are bound to host identities (HITs) and not IP addresses. This sub-layer is responsible for maintaining the binding between HITs and IP addresses. However, each host must also know at least one IP address at which its peers are reachable. Initially, these IP addresses are the ones used during the HIP base exchange [20].

Consider the case in which there is no mobility. The HIP base exchange establishes the HITs in use between the hosts, the Security Parameters Indices (SPIs) to use for ESP, and the IP addresses. There can only be one such set of bindings in the outbound direction for any given packet, and the only fields used for the binding at the HIP layer are the fields exposed by ESP (the SPI and HITs). For the inbound direction, the SPI is all that is required to find the right host context.

Consider a mobility event, in which a host moves to another IP address. Two events occur in this case. First, the peer must be notified of the address change using a HIP UPDATE message. Second, each host must change its local bindings at the HIP sub-layer which updates new IP addresses.

HIP requires a change in the transport layer sockets which in turn requires adaptation by applications in order to support mobility. As compared, TMSP continues the use of existing sockets to provide transparency of IP addresses due to mobility. This is accomplished by using the IP address used at establishment of connection as the host identifier.

Currently, HIP still requires support from existing infrastructure, including the extensions to DNS [21], and a new piece of infrastructure, called the HIP rendezvous server (RVS) [22]. The RVS serves as an initial contact point for HIP nodes. HIP nodes use the HIP Registration Protocol to register their HIT to IP address mappings with the RVS. After this registration, other HIP nodes can initiate a base exchange using the IP address of the RVS instead of the current IP address of the node they attempt to contact. In addition, more work is required to study the interactions of HIP with legacy NATs and legacy applications.

TMSP taps on the pervasiveness of SIP by using SIP URI as host identifier. Resolution of SIP URI to IP address can be automated by upgrading library functions like gethostbyname() and resolution of IP address to SIP URI can be obtained by trapping new connection to acquire SIP URI from the CN's SIP user agent. This supports execution of legacy applications.

B. Micro-Mobility

Hierarchical MIPv6 [23] proposes an extension for MIPv6 to enable micro-mobility support. It separates the crossing of AP with the same network domain (micro-mobility) and crossing to another network domain (global mobility). In this protocol, local handovers are managed locally and transparently to MN's CNs in other network domains. This protocol has at least two advantages. Since local handovers are performed locally, it increases the handover speed and minimizes the loss of packets during transition. Thus, it improves handover performances. It also reduces the signaling load on the Internet as the signaling messages corresponding to the local handover stay within the local domain.

A MN acquires a private CoA (PCoA) and a virtual CoA (VCoA) when it enters a HMIPv6 domain that is hosted by a mobility server (MS). A MS is a router that maintains a binding for each MN currently visiting the domain. MN will then send a signaling message (BU) that specifies a binding between its VCoA and PCoA to the MS. After that, it will send a MIPv6 BU that specifies the binding between its HoA and VCoA to its HA and its CNs that do not reside in the local domain. This ensures that packets destined for MN using its HoA will be routed to the MS, who in turn tunnels the packets to MN's PCoA. Finally, it will send a BU that specifies a binding between its HoA and its PCoA to its CNs that reside in the local domain. This allows MN's CN residing in the local domain to route packets directly to MN using MN's PCoA. To maintain these IP address bindings, MN will only need to update its newly acquired PCoA to the MS and CNs residing in the local domain.

TMSP can easily be adapted to support this micro-mobility extension. Similarily, MN sends the same BU that specifies a binding between its VCoA and PCoA to the MS. After that, it will register its VCoA with its SIP-RS, and re-invite all its CNs that do not reside in the local domain using its VCoA. Likewise, MN will re-invite all its CNs that reside in the local domain using its PCoA.

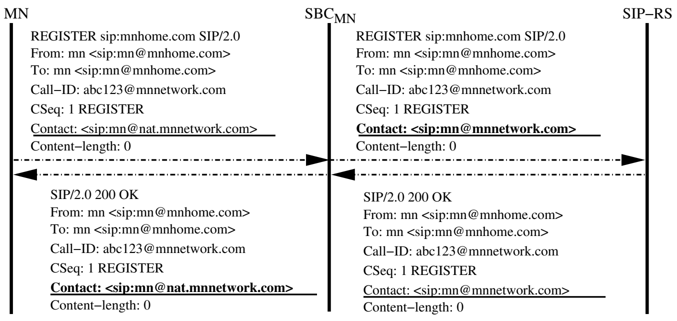
Further optimization work is introduced for hierarchical MIPv6 [24]. An adaptive route optimization algorithm is put forward to address a scenario that hierarchical MIPv6 may degrade end-to-end data throughput due to the tunnelling between MS and MN. This happens with a highly active MN with low mobility. Data throughput can be improved without using the MS tunnelling. The algorithm chooses the best scheme adaptively to improve the throughput performances based on measured session-to-mobility ratio (SMR) of MN. Similarly, TMSP can be adapted to utilize the benefits of SMR for adaptive route optimization.

C. Firewall and NAT traversal

A session border controller (SBC) is a proprietary network intermediary deployed at the border of a network to enforce network policies to provide a variety of functions to enable or enhance SIP services, like VoIP. Some of its common functions include perimeter defense like access control, topology hiding, denial-of-service prevention and detection, NAT traversal, and network management like traffic monitoring, shaping and QoS. Currently, SBC is not a standard solution but it has gained the attention of IETF who has begun to gather the requirements from SBC deployments [25]. A SBC enables SIP signaling and media to be received from and directed to a user device behind a firewall and NAT. It achieves this by rewriting the IP addresses and ports in the call signaling headers and the session description protocol blocks attached to SIP messages.

A SBC sits between MN and SIP registrar of the domain to perform NAT traversal function. A NAT traversal function modifies specific SIP messages to assist a MN in maintaining connectivity between its UA and SIP registrar, and, CNs' UA.

To illustrate, assume that MN and its CN are both in NAT hosted by their respective SBCs. As shown in



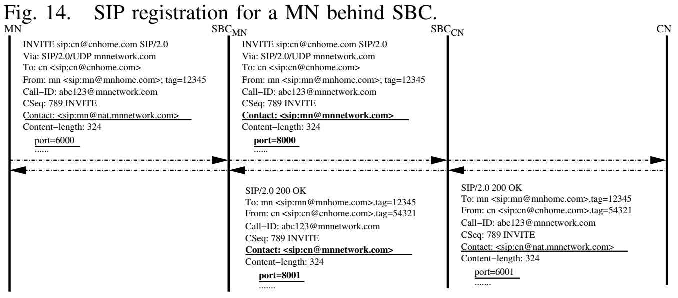
```
REGISTER sip:mnhome.com SIP/2.0
From: mn <sip:mn@mnhome.com>
To: mn <sip:mn@mnhome.com>
Call–ID: abc123@mnnetwork.com
CSeq: 1 REGISTER
Contact: <sip:mn@nat.mnnetwork.com>
Content–length: 0
```

```
REGISTER sip:mnhome.com SIP/2.0
From: mn <sip:mn@mnhome.com>
To: mn <sip:mn@mnhome.com>
Call–ID: abc123@mnnetwork.com
CSeq: 1 REGISTER
Contact: <sip:mn@mnnetwork.com>
Content–length: 0
```

```
SIP/2.0 200 OK
From: mn <sip:mn@mnhome.com>
To: mn <sip:mn@mnhome.com>
Call–ID: abc123@mnnetwork.com
CSeq: 1 REGISTER
Contact: <sip:mn@nat.mnnetwork.com>
Content–length: 0
```

```
SIP/2.0 200 OK
From: mn <sip:mn@mnhome.com>
To: mn <sip:mn@mnhome.com>
Call–ID: abc123@mnnetwork.com
CSeq: 1 REGISTER
Contact: <sip:mn@mnnetwork.com>
Content–length: 0
```

Fig. 14. SIP registration for a MN behind SBC.



```
INVITE sip:cn@cnhome.com SIP/2.0
Via: SIP/2.0/UDP mnnetwork.com
To: cn <sip:cn@cnhome.com>
From: mn <sip:mn@mnhome.com>; tag=12345
Call–ID: abc123@mnnetwork.com
CSeq: 789 INVITE
Contact: <sip:mn@nat.mnnetwork.com>
Content–length: 324
    port=6000
    ......
```

```
INVITE sip:cn@cnhome.com SIP/2.0
Via: SIP/2.0/UDP mnnetwork.com
To: cn <sip:cn@cnhome.com>
From: mn <sip:mn@mnhome.com>; tag=12345
Call–ID: abc123@mnnetwork.com
CSeq: 789 INVITE
Contact: <sip:mn@mnnetwork.com>
Content–length: 324
    port=8000
    ......
```

```
SIP/2.0 200 OK
To: mn <sip:mn@mnhome.com>.tag=12345
From: cn <sip:cn@cnhome.com>.tag=54321
Call–ID: abc123@mnnetwork.com
CSeq: 789 INVITE
Contact: <sip:cn@mnnetwork.com>
Content–length: 324
    port=8001
    .......
```

```
SIP/2.0 200 OK
To: mn <sip:mn@mnhome.com>.tag=12345
From: cn <sip:cn@cnhome.com>.tag=54321
Call–ID: abc123@mnnetwork.com
CSeq: 789 INVITE
Contact: <sip:cn@nat.mnnetwork.com>
Content–length: 324
    port=6001
    .......
```

Fig. 15. Mobility session establishment between MN and its CN via SBCs.

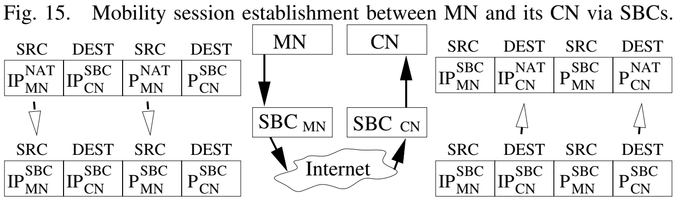

SRC DEST SRC DEST
IP^{NAT}_{MN} IP^{SBC}_{CN} P^{NAT}_{MN} P^{SBC}_{CN}
SRC DEST SRC DEST
IP^{SBC}_{MN} IP^{SBC}_{CN} P^{SBC}_{MN} P^{SBC}_{CN}
MN
CN
SBC $_{MN}$
SBC $_{CN}$
Internet
SRC DEST SRC DEST
IP^{SBC}_{MN} IP^{NAT}_{CN} P^{SBC}_{MN} P^{NAT}_{CN}
SRC DEST SRC DEST
IP^{SBC}_{MN} IP^{SBC}_{CN} P^{SBC}_{MN} P^{SBC}_{CN}

Fig. 16. Pathway of an IP packet from MN to CN passing through SBCs.

fig. 14, MN acquires a NAT address IP^{NAT}_{MN} (represented by "nat.mnnetwork.com") upon joining the network. Its SBC modifies MN's register request to its SIP-RS so that MN's SIP-RS registers the IP address of MN as the IP address of SBC, IP^{SBC}_{MN} (represented by "mnnetwork.com"). Likewise for CN, CN's SIP-RS registers CN's IP address as the IP address of CN's SBC.

Fig. 15 shows the exchanges SIP messages when MN initiates a mobility session to CN via SBCs. When MN calls CN, it resolves that CN is using IP^{SBC}_{CN} and a SIP invite request is sent to CN through CN's SBC who performs the NAT traversal and SIP message manipulation. The port number used by CN is reported in its response to MN's SIP invite request and is modified by CN's SBC who may use another port number. Thus, MN registers that the connection uses CN's SBC IP address and port number and CN registers that the connection uses MN's SBC IP address and port number. A SBC can use the SIP register request to refresh the binding of MN's NAT. To adjust the refreshing rate of the registration procedure to the expiry time required by SBC, it can modify the validity of registration (i.e., earlier registration expiring time) when MN's UA registers with SIP-RS.

With this establishment, the SBCs open the selected ports and create a NAT binding for this connection. A packet destined to CN from MN (see fig. 16) will use (IP^{NAT}_{MN},P_{MN}) as the source IP address and port number, and (IP^{SBC}_{CN},P^{SBC}_{CN}) as the destination IP address and port number. This packet reaches MN's SBC where it undergoes a NAT process. SBC replaces the source IP address and port number to (IP_{MN}^{SBC},P_{MN}^{SBC}) whereby IP_{MN}^{SBC} is the IP address of MN's SBC and P_{MN}^{SBC} is a port opened by SBC which maps to P_{MN}. Through conventional IP routing, this packet reaches CN's SBC where it undergoes another NAT process that translates the packet's destination IP address from public address to the private address used by CN. CN's SBC translates the destination IP address and port to (IP_{CN}^{NAT},P_{CN}) and forwards this packet to CN.

Similar to micro-mobility, when MN moves, it updates its SIP-RS of its new IP address. The SBC receives this SIP register message, updates its NAT binding for MN, and modifies the IP address reported in the register message to its IP address. Thus, to the SIP-RS, the MN did not move. Likewise, SBC performs similar modification for each SIP invite request message sent by MN to update its CNs.

To support movement of MN to a public network or another NAT traversal network, TMSP needs to perform port number mapping in addition to the IP address mapping for each connection. In the case when a MN moves from a NAT traversal network to a public network, its UA sends a non-standard SIP invite request to its CNs in which it requests CNs to report the port numbers previously used by MN as recorded in CNs' state tables. With this information, MN records the port numbers accordingly and maps both the IP addresses and the ports for each incoming packet in its AMM module. In the case when a MN moves from a NAT traversal network to another, the SBC in the new network parses all the SIP register and invites requests. Each invite request message to CN includes all the ports currently used to listen to packets from CN. SBC appends a non-standard tag onto the message which includes proposed port numbers for each MN's port to request CN's UA to map outgoing packets to MN using this port number. Upon obtaining a response from CN, SBC opens the port on the firewall. With a mapping of outgoing packet request, CN maps the IP addresses and port numbers for each outgoing packet.

VIII. Conclusion

TMSP enables IP mobility for MNs. It creates a association of IP addresses and port numbers for a connection in MN and its CNs, and uses this association to hide the changes in IP address when MN moves from one network to another. Owing to the manipulation of IP addresses in the IP header of each packet, TMSP ensures that packets are sent directly between CN and MN without packet redirection. TMSP taps on the pervasiveness of SIP server as an IP directory service and uses SIP as a signaling mechanism to maintain location information. Each MN is identified by a SIP URI. Thus, TMSP does not require permanent IP address for each MN, does not require new network infrastructure support, and does not enforce packet redirecting during routing. Additionally, TMSP works for IPSec and the operation of TMSP with firewall and NAT using SBC is described. Through quantitative analysis, TMSP outperforms MIPv6 and SIP-MIPv6 in terms of overheads in signaling and data transmission. Lastly, TMSP has been implemented and shown that it works for UDP and TCP connections. Furthermore, it requires little processing time and does not affect the performances of the wireless LAN bandwidth.

References

[1] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler, "SIP: Session Initiation Protocol," RFC 3261, June 2002.

[2] E. C. Perkins, "IP Mobility Support for IPv4," RFC 3344, Aug. 2002.

[3] D. Johnson, C. Perkins, and J. Arkko, "Mobililty Support in IPv6," RFC 3775, June 2004.

[4] D. Harkins and D. Carrel, "The Internet Key Exchange (IKE)," RFC 2409, Nov. 1998.

[5] H. Schulzrinne and E. Wedkund, "Application-layer mobility using SIP," Mobile Computing and Communications Review, vol. 1, no. 2, pp. 47–57, July 2000.

[6] C. Perkins, "IP Encapsulation within IP," RFC 2003, Oct. 1996.

[7] S. Kent and R. Atkinson, "IP Authentication Header," RFC 2402, Nov. 1998.

[8] ——, "IP Encapsulating Security Payload (ESP)," RFC 2406, Nov. 1998.

[9] P. Ferguson and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which empoly IP Source Address Spoofing," RFC 2267, Jan. 1998.

[10] E. Wedkund and H. Schulzrinne, "Mobility support using SIP," in ACM WoWMoM, Seattle, USA, Aug. 1999, pp. 76 – 82.

[11] A. Fei, G. Pei, R. Liu, and L. Zhang, "Measurements on delay and hop-count of the internet," in Proc. IEEE International Conference on Global Telecommunications, Sydney, Nov. 1998.

[12] A. C. Snoeren and H. Balakrishnan, "An end-to-end approach to host mobility," in ACM/IEEE International Conference on Mobile Computing and Networking, Boston, MA, Aug. 2002.

[13] G. Bianchi, "Performance analysis of the IEEE 802.11 distributed coordination function," IEEE J. Select. Areas Commun., vol. 18, no. 3, pp. 535 – 547, Mar. 2000.

[14] B. Aboba, J. Carlson, and S. Cheshire, "Detecting Network Attachment in IPv4 (DNAv4)," RFC 4436, Mar. 2006.

[15] S. Park, P. Kim, and B. Volz, "Rapid Commit Option for the Dynamic Host Configuration Protocol version 4 (DHCPv4)," RFC 4039, Mar. 2005.

[16] C. R. and B. P., "MultiNet: connecting to multiple IEEE 802.11 networks using a single wireless card," in Proc. IEEE INFOCOM, vol. 2, Hong Kong, Mar. 7–11, 2004, pp. 882–893.

[17] E. R. Koodli, "Fast Handovers for Mobile IPv6," RFC 4068, July 2005.

[18] M. Buddhikot, A. Hari, K. Singh, and S. Miller, "Mobilenat: A new technique for mobility across heterogeneous address space," in ACM International Workshop on Wireless Mobile Applications and Services on WLAN Hotspots, San Diego, California, USA, Sept. 2003.

[19] P. Srisuresh, J. Kuthan, J. Rosenberg, A. Molitor, and A. Rayhan, "Middlebox communication architecture and framework," RFC 3303, Aug. 2002.

[20] S. Moskwitz, P. Nikander, P. Jokela, and T. Henderson, "Host identity protocol," Internet Draft, Oct. 2007, working-in-progress, draft-ietf-hip-base-10.

[21] P. Nikander and J. Laganier, "Host Identity Protocol (HIP) Domain Name System (DNS) Extensions," Internet Draft, Apr. 2007, draft-ietf-hip-dns-09.txt, Working-in-progress.

[22] J. Laganier and L. Eggert, "Host Identity Protocol (HIP) Rendezvous Extension," Internet Draft, June 2006, draft-ietf-hip-rvs-05.txt, Working-in-progress.

[23] H. Soliman, C. Castelluccia, K. E. Malki, and L. Bellier, "Hierarchical Mobile IPv6 Mobility Management (HMIPv6)," RFC 4140, Aug. 2005.

[24] S. Pack, X. Shen, J. w. Mark, and J. Pan, "Adaptive route optimization in hierarchial mobile ipv6 networks," IEEE Trans. Mobile Computing, vol. 6, no. 8, pp. 903–914, Aug. 2007.

[25] J. Hautakorpi, G. Camarillo, R. Penfield, A. Hawrylyshen, and M. Bhatia, "Requirements from SIP (Session Initiation Protocol) Session Border Control Deployments," Internet Draft, Nov. 2006, draft-ietf-sipping-sbc-funcs-00.txt, Working-in-progress.

Teck Meng Lim received his BASc (Hons) and PhD from the School of Computer Engineering, Nanyang Technological University in 2000 and 2005, respectively. Currently, he is a Technology and Product Manager in StarHub Ltd, Singapore. His research interests include mobility management, network QoS, wireless LAN, ad hoc network, pervasive communication and Internet technologies.

Chai Kiat Yeo received the B.E. (Hons.) and M.Sc. degrees in 1987 and 1991 respectively, both in electrical engineering, from the National University of Singapore and the Ph.D. degree from the School of Electrical and Electronics Engineering, Nanyang Technological University (NTU), Singapore, in 2007. She was a Principal Engineer with Singapore Technologies Electronics and Engineering Limited prior to joining NTU in 1993. She has been the Deputy Director of Centre for Multimedia and Network Technology (CeMNet) in Nanyang Technological University (NTU), Singapore. She is currently Associate Chair (Academic) with the School of Computer Engineering, NTU. Her current research interests include ad hoc and mobile networks, overlay networks, speech processing and enhancement.

Francis, Bu Sung Lee received the B.Sc. (Hons.) and Ph.D. degrees from the Electrical and Electronics Department, Loughborough University of Technology, U.K., in 1982 and 1987, respectively. He is currently Associate Chair (Research) with the School of Computer Engineering, Nanyang Technological University, Singapore. He was elected the inaugural President of Singapore Research and Education Networks (SingAREN) and has been an active member of several national standards organizations, including the National Grid Pilot Project and international organizations such as Asia Pacific Advanced Networks (APAN). His research interests include computer networks protocols, distributed computing, network management and grid computing.

Quang Vinh Le is currently working as Consultant at Fiserv, a Fortune 500 company. His area of expertise is banking applications. Before joining Fiserv, he worked as Solutions Consultant in Catena Technologies, specializing in technological financial services. Between 2005 and 2006, Vinh worked at the Nanyang Technological University, CeMNet as Research Officer, specializing in wireless mobility. He graduated from the National University of Singapore, School of Computing with an Honours Degree in Computer Science in July 2005.