Georg-August-Universität
Göttingen
Institut für Informatik

ISSN 1611-1044
Nummer IFI-TB-2006-04

Technischer Bericht

TCP-M6: Mobility Aware TCP Extensions For Mobile IPv6

Deguang Le, Xiaoming Fu, Dieter Hogrefe

Technische Berichte
des Instituts für Informatik
an der Georg-August-Universität Göttingen

March 2006

Georg-August-Universität Göttingen
Zentrum für Informatik

Lotzestraße 16-18
37083 Göttingen
Germany

Tel.	+49 (5 51) 39-1 44 14
Fax	+49 (5 51) 39-1 44 15
Email	office@informatik.uni-goettingen.de
WWW	www.ifi.informatik.uni-goettingen.de

TCP-M6: Mobility Aware TCP Extensions for Mobile IPv6

Deguang Le, Xiaoming Fu, Dieter Hogrefe
Institute for Informatics, Universität Göttingen
Lotzestr. 16-18, Göttingen 37083, Germany
Email: {le,fu,hogrefe}@cs.uni-goettingen.de

***Abstract*— To provide higher layer transparency, Mobile IPv6 (MIPv6) hides mobility from transport layer such as TCP. However, it poses substantial impacts on TCP due to mobility issues, including packets loss, variation of packet size, and deviation of end-to-end transport delay, which may seriously degrade the TCP transport performance. This paper explores the impact of MIPv6 on TCP and proposes a scheme for MIPv6-aware TCP extensions by introducing a mobility-detection network layer module and a mobility-aware transport layer module. The changes are only made to the endpoints and preserve the end-to-end semantics of TCP. Different from most exiting works, which utilize either transport or network layer alone without much consideration on inter-layer collaboration, our approach uses MIPv6-related information to perform reactive TCP operations. Through performance simulations, our approach demonstrates that it can significantly improve TCP transport performance in MIPv6-based mobile computing environments, which outperforms prior approaches.**

***Index Terms*— Mobility Control, MIPv6, TCP**

I. Introduction

With the wide range of wireless access technologies, such as IEEE 802.11x, GPRS, Third Generation (3G) cellular networks, and Bluetooth etc., available, the increasing wireless networks have been developed and implemented, building into the heterogeneous wireless networks [1]. Besides, with the fast evolution of mobile communication and Internet technologies, there is a strong need to provide connectivity for moving devices to communicate with other devices on the Internet. Internet mobility support has been a hot topic in the last decade, and studies that address this issue have arisen, coming up with a number of protocol proposals and schemes [2]. Among them, Mobile IPv6 (MIPv6) [3] as the most mature solution has been supported and adopted by mobile devices and network equipment vendors [4], [5], and some of network providers are even starting to deploy the MIPv6 networks [6]. Future Internet will consist of different heterogeneous wireless networks uniformed with a common MIPv6 platform.

The MIPv6 presents the notion of network layer mobility management. Its basic principle is splitting the traditional dual roles of IP identifier and locator by introducing a so-called Care of Address (CoA) as the locator of host, which allows a Mobile Node (MN) to move from one network to another without changing the MN's home address (i.e. act as host identifier) [7]. Packets thus may be routed between the MN and the Correspondent Node (CN) regardless of the MN's current point of attachment to the Internet.

The splitting mechanism handled at the network layer makes the transport layer transparent for the mobility support. However, the complete transparency may not be achievable especially for the motion-unaware Transmission Control Protocol (TCP) [8], [9], [10], [11], [12].

As we know, TCP [13] is a reliable transport protocol tuned to perform well in traditional fixed networks, where the route for each active TCP connection is relatively invariable and network congestion is the primary factor of affecting the TCP performance. However, in modern mobile networks, it is common that the route of active connection changes due to communicating endpoints moving over the heterogeneous networks with different network features, which incurs significant end-to-end transport delay deviation, packet size variation, and packet loss due to networks switching and mobile packet routing. For traditional motion-unaware TCP, it may consider these adverse effects induced by mobility issues as network congestion and trigger congestion control procedures (e.g. exponential backoff of retransmission timeout or reduction of its congestion window) mistakenly, causing transport performance degradation seriously.

Since mobile services are the future main form provided on the MIPv6-based Internet and TCP is the most widely used transport protocol, it is desirable for the changes of TCP to handle mobility control in MIPv6 to adapt itself to the mobile Internet.

Many researchers have done many works to evaluate transport performance in mobile environments and several approaches have been proposed for improving the transport performance [14], [15], [16], [17], [18], [19], [20], [21], [22], [23]. Some of them concentrated on the problem associated with wireless link features such as high bit error rate; some other researchers were interested in other limitations of mobile environments like the disconnection; and some other researchers presented the survey of the research done to improve the TCP performance over mobile networks. Based on the predecessors' work, in this paper, we quantify the effects of mobility on the TCP performance including overhead, transport delay and throughput etc., identify the factors that contribute to the degradation of TCP performance through measuring TCP behaviours in MIPv6, and propose the mobility control mechanisms through the enhancement of cooperation between the network and transport layers, and performing reaction adaptively, which can adapt TCP and MIPv6 protocols to perform better in mobile environments.

We also present an investigation of our scheme by means of simulation, where the TCP behaviours as well as performance are evaluated. The simulation results shows that the our proposed approach can yield better performance. In comparison with previous works, the contributions of this work include:

- Firstly,we propose the inter-layer enhancement to enable TCP aware of mobility-related behaviours, so that the mobility-aware TCP can perform better for improvement of transport performance in mobile environments.
- Secondly, most previous works only deal with packet loss issue induced by lossy wireless link or disconnection. In our work, we widely analyze the possible problems induced on the mobile Internet including not only the packet loss but also the end-to-end transport delay deviation and the packet variation. And we solve these problems through our TCP mobility control mechanisms.
- Thirdly, most previous works only assume the scenario where the TCP sender is the fixed node while the mobile node acts as the TCP receiver. Thus, their corresponding improving approaches were only proposed to deal with the scenario. However in mobile environments, it is common that the mobile node may act as the sender of TCP like in the more and more popular point-to-point applications. Therefore, in our paper, we analyze the more general scenario, where both communicating endpoints may be the sender of TCP, and propose corresponding operations to improve the transport performance.
- Fourthly, most of current works don't consider specific mobility management scheme while they study on this field. As we know, MIPv6 is one of the most mature mobility management schemes, so we combine our work with the most popular mobility management scheme, which make our work have more practical significance.
- Finally, the overall transport performance in our scheme has been improved including overhead as well as throughput.

The remainder of this paper is organized as follows. Section 2 presents some background about the MIPv6 technologies and the TCP overview. And we discuss the related work in Section 3. In Section 4, we analyze the problems of TCP related to mobility in MIPv6 . Section 5 describes our proposed scheme in detail. In Section 6, we present the implementation of this new protocol stack model based on the OPNET simulator, describe the simulation scenario and measure and evaluate simulation results. Finally, conclusions are provided in Section 7.

II. Background

In this Section, we briefly describe the MIPv6 technologies including handover mechanisms and mobile packet routing mechanisms, which may affect the TCP behaviours. Then we introduce the transport protocol: TCP and its features, which guarantee high transport performance in the traditional fixed networks, including the acknowledgement/retransmission, congestion control, timing, and maximum segment transmission mechanism etc.

A. MIPv6 Technologies

In MIPv6, the handover mechanisms can be regarded to comprise three procedures: movement detection, CoA configuration, binding registration.

- Movement Detection: When a MN moves, it must detect its current location. In MIPv6, a MN can determine its current location by listening to the router advertisements and comparing the network prefix of the source address within this advertisement with the network prefix of its home network. If the network prefix of the source address within the router advertisement equals the network prefix of the home address of MN, then the MN is on its home network. Otherwise the MN is on a foreign network.
- CoA Configuration: To obtain a CoA, the MN can use either stateful or stateless address auto-configuration methods. In the first situation, the MN obtains a CoA from a Dynamic Host Configuration Protocol for IPv6 (DHCPv6) [24] server. In the latter situation, by using the Neighbour Discovery protocol [25], a MN is able to find the network prefix at any point of attachment, and then adds a unique interface identifier to form a CoA for that point of attachment.
- Binding Registration: When a MN moves to a foreign network and acquires a CoA, it then registers to the HA or CNs in its list to inform them of its current location by using the IPv6 Mobility header [3]. The MN performs the binding registration by sending a Binding Update message to the HA or CNs. The HA or CNs reply to the MN by returning a Binding Acknowledgement message.

In order that the communicating endpoints (i.e. the MN and CN) can trace and route packets each other continuatively even though movement, MIPv6 also specifies two routing mechanisms for packet transmission between the MN and the CN: Bidirectional Tunnelling (BT) and Route Optimization (RO). Figure 1 illustrates the mobile routing mechanisms in MIPv6.

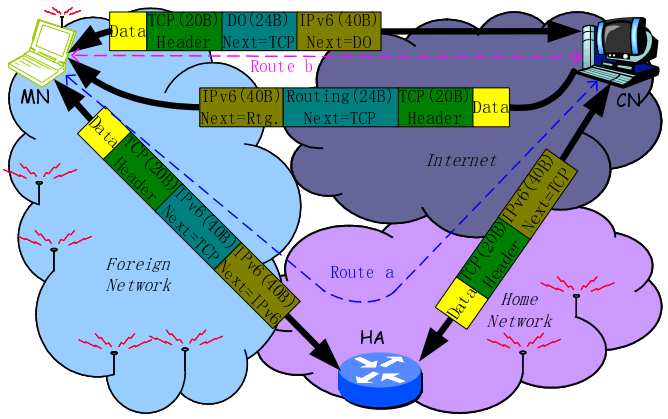

Fig. 1. The mobile packet routing mechanisms in MIPv6

In the BT mechanism, packets from the CN to the MN are routed to the home address of MN, the HA shall uses proxy Neighbour Discovery [25] to intercept any IPv6 packets

addressed to the MN's home address on the home network. Each intercepted packet is tunnelled to the MN's current CoA [26]. Packets to the CN are tunnelled from the MN to the HA, which is called reverse tunnelling, and then routed normally from the home network to the CN (see figure 1, route a).

In the RO mechanism, the HA no longer exclusively deals with the address mapping, but each CN can have its own binding cache. In the direction from the MN to the CN, packets sent by the MN are delivered to the CN with the Home Address option in the Destination Option Extension header when the MN is away from its home network. In this case, the MN sets the IPv6 header's source address to its CoA and adds a Home Address option with the MN's home address to the IPv6 header. When the CN receives the packet from the MN, it replaces the MN's home address to be the IPv6 header's source address before delivering the packet to the upper layer. In the opposite direction, when sending packets to the MN, the CN checks its cached bindings for an entry for the packets' destination address. If a cached binding entry for this destination address is found, the CN uses the Type 2 Routing header to route packets to the MN by specifying the CoA as the destination address in the IPv6 header and the MN's home address as the final destination in the Routing header (see figure 1, route b). If no cached binding entry for this destination address is found, for examples the binding is timeout or the CN originates a communication with the MN etc., the CN does not know the current location of MN. In this case, it shall send packets using the MN's home address as destination address. The MN's HA will intercept the packets and tunnels them to the current location of MN (see figure 1, route a). When the MN receives packets from its HA, it is aware that the CN does not know its current CoA and will inform the CN of the current CoA by sending a Binding Update message to the CN, so that the CN later can send packets to the MN directly.

B. TCP Overview

TCP [13] is the most widely used transport layer protocol for its reliability and high performance, which is achieved with the acknowledgement, retransmission, congestion control, timing, maximum segment transmission mechanisms. In this Subsection, we will introduce these features that characterize the quality of services.

1) Acknowledgment and Retransmission: In TCP, each communicating endpoint acknowledges the segment it receives from the other endpoint. In practice, TCP employs cumulative acknowledgements that acknowledge the reception of all segments that were sent before the present segment.

However, both data segments and acknowledgements can get lost. TCP recovers the lost data segments by the retransmission mechanism. Acknowledgement and Retransmission together guarantee TCP to provide the reliable delivery service.

2) Timing: To handle retransmission, TCP sets a retransmission timeout (RTO) when it sends data. And if the data is not acknowledged when the timeout expires, it retransmits the data. The value of RTO is dynamically computed based on the estimated Round Trip Time (RTT) and its mean deviation through a smooth algorithm [27].

When a segment is retransmitted, the retransmission timer is set to a backoff interval that doubles with each consecutive timeout according to Karn's exponential timer backoff algorithm [28]. This prevents retransmission from being sent too quickly and further overloading the network.

3) Congestion Control: It is common that packet loss may be introduced in packet-switched networks for the reason of network congestion. TCP adapts itself to the problem by the congestion control mechanism. The congestion control mechanism is based on assuming that every lost packet, for which the sender does not receive an acknowledgement, has been dropped due to an overloaded network. Following this assumption, the sender will reduce its throughput until complete network recovery by introducing two parameters in the TCP sender: the congestion window (CWND) and the slow start threshold (SSTHRESH). In the traditional TCP implementation, the congestion control mechanism is based on the following algorithms: slow start, congestion avoidance, fast retransmission, and fast recovery [29], which are independent and follow different objectives. Figure 2 shows the TCP CWND/SSTHRESH grows and varies with congestion algorithms.

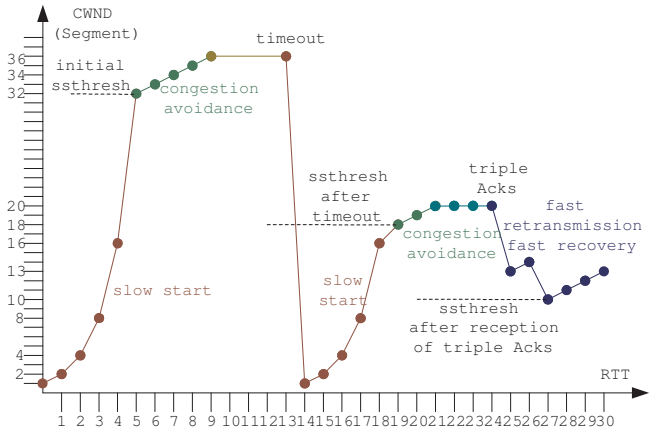

Fig. 2. Visualization of congestion control mechanisms

4) Maximum Segment Transmission: TCP segments are the messages that carry actual data between TCP devices. In TCP/IP-based networks, each delivered packet includes protocol headers. Among them, the TCP header takes up 20 bytes of data (or more if options are used); the IPv6 header also uses 40 or more bytes. This means that between them a minimum of 60 bytes are needed for headers, all of which is non-data overhead. If we set the length of TCP segment too low, this results in very inefficient use of bandwidth. Therefore the larger the TCP segment the better for the sake of overhead. However, this may not always be true. Because packets have their own size limit issue with the matter of the Maximum Transmission Unit (MTU) of an underlying network, if a TCP segment is too large, it will result that the IP datagram is too large to be sent without fragmentation. Fragmentation reduces efficiency and increases the chance of part of a TCP segment being lost, causing the entire segment needs to be retransmitted.

To balance these two competing issues, TCP employs a parameter to restrict the size of the segment it sends, called Maximum Segment Size (MSS). The MSS is chosen as large as possible while avoiding fragmentation for transmitted segments. The standard TCP MSS is computed by starting with the outgoing link's MTU, minus the size of the TCP and IP headers. Besides, TCP also specifies a MSS option to inform each other of endpoints of the MSS it wants to use during the connection establishment process.

III. Related Work

From the above presentation about the background, we can see that the transport layer has no feature that considers mobility issues treated at the network layer, which may result in the performance degradation, especially in the case of reliable end-to-end communication. It is therefore necessary to implement the TCP protocol for mobile environments, which will provide the MN with the same services as offered to fixed nodes without degrading the transport performance while providing the mobility support. A number of studies for this issue have been conducted and numerous schemes have been proposed in the literature. The proposed schemes can be broadly classified into those that operate at the network layer [14], [15] and those that operate at the transport layer [16], [17], [18], [19], [20], [21], [22], which will be presented in the following subsections.

A. Considerations at the Network Layer

Some literature like [8] shows that although network layer mobility management may be appropriate for current applications, its long handover periods make it unsuitable for the future in term of the transport performance. As we will explore in Section 4, the mobility management of MIPv6 at the network layer has an adverse effect on the TCP. To improve the TCP performance in MIPv6, one apparent way is to improve the inefficiency of MIPv6 handover procedures. And IETF develops several proposals like the Hierarchical Mobile IPv6 Mobility Management (HMIPv6) [14] and Fast Handovers for Mobile IPv6 (FMIPv6) [15] etc. to address this issue. Essentially, the network layer schemes attempt to minimize the registration latency by introducing a hierarchical structure, and lower the address configuration latency through advanced configuration. And Hsieh et al. [30], [31] present a performance analysis and comparison of the current IETF proposals for end-to-end TCP.

IETF proposes the HMIPv6 protocol for local mobility support to improve the handover speed and to reduce the packet loss by separating the local mobility from the global mobility. HMIPv6 introduces a n-level hierarchical network architecture and defines a site as the highest level of the hierarchical architecture. Inside the site a new entity called Mobility Anchor Point (MAP) is introduced. It acts like a local HA, when the MN moves within the site, the MN shall only register its new local CoA by sending the BU to the MAP. The local mobility can be completely hidden from all nodes outside the site. When the MN moves between inter-sites, the mobility shall be handled by the standard MIPv6.

FMIPv6 is an alternative optimization for MIPv6. The main idea of FMIPv6 is to attempt to acquire information that is needed to join a new network before disconnecting communication at the old network. It utilizes co-operating access routers which can request information from other access routers that are possible candidates for a handover. This is done by establishing a tunnel between the two access routers that allows the MN to send packets as if it was connected to its old access router while it is completing its handover procedures at its new access router.

These schemes have improved the handover latency and reduced the impact of mobility on TCP to some extent. However, the overall improvement of TCP performance is not achievable simply. [32] shows that the pure extensions to MIPv6 through only improving handover latency does not necessarily/certainly improves the performance of TCP in term of the throughput. Another disadvantage of these schemes is their operational complexity such as packets forward through tunnelling between access routers etc., which induces significantly deviation of end-to-end packet transport. In particular, in heterogeneous network, where the MN may perform the vertical handover, the change of link characteristic (like network bandwidth etc.) and propagation delay may erroneously invoke congestion control procedures and thus degrade the TCP performance [33]. In addition, for the variation of packet size, the only way these IP layer schemes can deal is to divide packet into smaller packets through IP fragmentation/reassembly mechanisms, which deteriorates the transport performance.

B. Considerations at the Transport Layer

The transport layer schemes are orthogonal to The above network layer schemes. For example, Balakrishnan et al. [17], [18] design and implement a snoop module to improve the reliable transport and the handover performance in cellular wireless networks; Bakre et al. [19] describe an indirect transport protocol (I-TCP) for mobile hosts; Brown et al. [20] develop the M-TCP for use in mobile networks; Caceres et al. [16] propose an end-to-end fast retransmission scheme to reduce unacceptably long pauses due to cellular handovers to levels more suitable for the human interaction; Stangel et al. [21] introduce a Mobile TCP to maintain robust operation despite the lossy mobile activity; and Goff et al. [22] optimize TCP for mobility through Freeze-TCP: a end-to-end TCP enhancement mechanism for mobile environments.

In [17], [18], a snoop module is added on an intermediate node between communicating endpoints such as the base station near the mobile endpoint. It monitors every segment on the connection in either direction, and caches the segments that are sent by the fixed endpoint to the mobile endpoint but have not yet been acknowledged by the mobile endpoint. On receiving a segment from the fixed endpoint, the module stores the segment in its cache and then passes it to the mobile endpoint. If the segments are lost on the wireless link, then the snoop-aware node gets repeated acknowledgements for the lost segments from the mobile endpoint, and then the snoop module checks if it has the segment in the cache: if it has

the segment, then retransmits the segment and suppresses the ACK to the fixed endpoint, otherwise forwards the ACK to the fixed endpoint and lets the sender recover from the loss.

I-TCP [19] works by breaking the connection between the fixed endpoint and the mobile endpoint in two connections. One connection is between the fixed endpoint and the base station, and the other connection is between the base station and the mobile endpoint. When a data segment sent to the mobile endpoint is first received by the base station, it sends an acknowledgement to the fixed endpoint and the data segment is forwarded to the mobile endpoint. The connection between the base station and the mobile endpoint needs not use traditional TCP for communication instead a specialized protocol that is optimized for mobile applications can be used. This indirection helps shield the wired network from the uncertainties of the wireless network and the TCP/IP at the fixed endpoint side needs not be changed. If the mobile endpoint moves to a different cell while communicating with a fixed endpoint, the whole connection information is transferred from the current base station to the new base station and the new base station takes over from here. The fixed endpoint is unaware of this indirection and is not affected when this switch occurs.

Similar to I-TCP, M-TCP [20] also uses the split connection mechanism. And it proposes a new three-level hierarchy by introducing a Supervisor Host (SH) for controlling several mobile support base stations, where the SH and the mobile endpoint communicate using specific M-TCP. In addition, M-TCP introduce timer freezing mechanism for preventing from the invocation of congestion control. In M-TCP, when the SH receives a data segment from the sender of fixed endpoint, it forwards the data segment to the mobile endpoint but defers the acknowledgement to the sender until it receives an acknowledgement from the mobile endpoint. Now at this point if the mobile endpoint undergoes a handover or a period of data loss, the SH sends the deferred acknowledgement and advertises the window size of zero, this causes the sender to go in a persist state. In this state all timers are frozen and does not close its congestion window and does not back off its timers. When the mobile endpoint regains the connection, it sends a greeting segment to the SH. SH-TCP sends a duplicate acknowledgement with the window update segment to the sender so that it can resume transmitting data.

The above schemes are based on the intuition that as the problem is local it should be solved locally and the TCP should be independent of the behaviour of the individual links. They hide the non-congestion related losses from the TCP at the fixed endpoint by introducing the intermediate node for substantial modifications. However, the feature of requirement of third devices makes it difficult for these schemes to inter-operate with the existing infrastructure with different administrative autonomies, where intermediate nodes are likely to belong to other organizations, making them unavailable for modifications in practice. And it is clear that the intermediary in SNOOP, I-TCP, M-TCP will all have to buffer at least some amount of data and do some extra processing for each connection going through them, which leads that the intermediary will become the bottleneck, even the loss of some segments and the sender dropping the congestion window, which would defeat the original purpose behind the whole endeavour. In [22], it was observed that SNOOP, I-TCP, M-TCP handle packet loss problem due to BER feature of wireless link well but do not effectively deal with the packet loss problem due to the handover. In addition, they don't consider the problem of packet delay deviation due to differences in/of network characteristics upon mobility/handover/motion.

Some other researchers try to enable the TCP aware of the existence of the wireless link and mobility in mobile environments, and to distinguish the impact due to mobility on the wireless link from other factors like the network congestion.

In [16], Caceres et al. make TCP to differentiate between the motion-related and congestion-related packet loss, which overcomes this problem by making the communicating endpoints resume the communication immediately after the handover without waiting for the timeout at the sender, called Fast Retransmission. In Fast Retransmission, the mobile endpoint sends triple duplicate ACKs to the sender. This prompts TCP at the sender to reduce the window size to half and begin retransmission.

Similar to Fast Retransmission, Mobile TCP [21] distinguishes the packet losses due to the handover and those due to the interface switching. Namely, it lets the MN tell the CN whether the loss is due to handover in the same network or if it is due to the interface switching. The sender then marks the packets and retransmits them once the MN has completed the handover. In case of the interface switch, the MN enters a new network, which may not have the same network characteristics as the previous one. When the TCP at the sender knows about the interface switching, it resets the window size and SSTHRESH, estimates of RTT and RTO values, and begins slow start. But if the MN has moved to a cell in the same network then the values of window size and SSTHRESH are halved and the RTT value remains the same.

Like M-TCP, Freeze TCP [22] is to enhance TCP over mobile environments by letting the communicating endpoints enter persist mode prior to a disconnection through signal strength measurements at the wireless antenna without the involvement of any intermediaries. In this protocol, when an MN detects impending handover, it tries to send out a few acknowledgements, wherein its window size is advertised as zero called Zero Window Advertisement (ZWA) to the CN. When the CN receives the ZWA, it freezes all retransmission timers and enters persist mode without shrinking its CWND size. As soon as the connection is reestablished, the MN sends three successive ACKs for the last data segment it received prior to the disconnection called Triplicate Re-connection ACKs (TR-ACKs) as suggested in [16]. When the CN receives them, it resumes transmission. This scheme does not require any help from any intermediate node and emphasizes that only the MN's TCP needs to be changed. Furthermore, it can be used with encrypted data.

Unlike SNOOP, I-TCP, M-TCP, which employs third devices for TCP improvement, the schemes of Fast Retransmission, Mobile TCP and Freeze TCP improve TCP at the communicating endpoints. In these above schemes, the enhanced TCP at endpoints is aware of the existence of the

wireless link in the network and is able to distinguish the packet loss due to mobility or transmission error over wireless link from those due to the network congestion, and responses through different mechanisms. Therefore, they can maintain the end-to-end semantics as well as no requirements of third devices. However, these end-to-end schemes only consider the scenario, where the fixed node is the tcp sender and the mobile node is the TCP receiver. With the growth of mobile users and diversification of application services, it will be more and more popular that the mobile node may also act as the sender of TCP such as in the point-to-point application. Therefore, it is necessary to differentiate and handle the two scenarios (them) respectively. For handle the packet loss problem due to the handover of mobility, in [21], although authors shows the packet loss in both scenarios, they do not handle them respectively. And these schemes mainly focus on (handle) the packet loss problem due to the handover. In [16], it simply retransmits the earliest unacknowledged segment immediately followed by the slow start or fast retransmission, which can reduce the unnecessary pauses for a timeout expiration. However, it does not prevent from invoking the congestion control procedure, which unnecessarily throttles the transmission window, still causing the throughput degradation when handover occurs consecutively. In Mobile TCP, it marks the data in its retransmit queue, so the marked packets can be retransmitted while not trigger the corresponding congestion control procedures. However, since it responses only after the handover completes, if the timeout expires before the handover completes, the slow start is still invoked. Freeze TCP uses the advanced detection mechanism for response. However, with Freeze TCP, a critical problem is that how much in advance of the disconnection the MN should start advertising a window size of zero, i.e. the warning period before which a actual disconnection happens. In fact, the performance improvement of this scheme is totally dependent on the accurate prediction of disconnection by the MN [34]. In [22], a reasonable warning period is estimated as the current RTT. If the warning period is any longer, the sender will go into persist mode too early, which leads to longer idle time and throughput degradation. On the other hand, if the warning period is too short, there may not be enough time for an MN to send the ZWA to the sender, which causes the sender's CWND size to be reduced in response to dropped packets during disconnection. According to [35], since the RTT is often measured very coarsely (the granularity is normal 500msec) by the sender instead of the MN, the assumption that the MN has the knowledge of RTT may not be practical. In addition, they also don't consider the problem of packet delay deviation due to differences in/of network characteristics upon mobility/handover/motion.

C. Comparisons

In this subsection, we will make a comparison to clarify the advantages of our scheme over previous work. Table I summarizes the problems handled by the schemes discussed in this paper and their features. The last column refers to our TCP-M6 scheme proposed in this paper.

From the table and our previous discussion, we can conclude as follows:

The end-to-end semantics is a remarkable feature of TCP. Like most of previous schemes, our TCP-M6 maintains the end-to-end semantics of TCP. For control localization, HMIPv6 introduces MAP for local registration, FMIPv6 employs access router for packet forwarding, or Snoop, I-TCP and M-TCP etc. employ base station for local handling. Our TCP-M6 does not depend on any other third devices, which does not destroy current network infrastructure. We also can see that Snoop and I-TCP handle wireless link bit error well but do not effectively deal with the packet loss due to the handover of/in mobility, and another good literature to optimize the wireless link bit error for the transport layer can be found in [36]. Therefore, our scheme mainly focuses on the more direct mobility-related features like the handover and traffic delay deviation etc.

In comparison with Fast Retransmission, Mobile TCP, and Freeze TCP, our TCP-M6 is based on mobile IPv6 environments, in which, we not only analyze the concrete reason for packet loss due to mobility/handover and propose mobility aware TCP extensions for MIPv6 through inter-layer enhancements and dynamical/adaptive/according TCP tuning, but also we they consider the problem of end-to-end transport delay deviation due to differences in/of network characteristics upon mobility/handover/motion. And more significantly, our scheme also can solve/eliminate the effect of packet size variation due to MIPv6 mobility management on the transport performance. (our TCP-M6 has the characteristics of adapting the packet size variation due to MIPv6 mobility protocol/management, which the previous schemes are short of.) In addition, although Mobile TCP [21] mentions/considers both scenarios that both endpoints may act as the sender of TCP, it does not handle them respectively, whereas it is observed that our TCP-M6 in the paper can deal with the scenario, where the MN acts as the sender of TCP, beside the traditional scenario where the fixed node is the sender of TCP. We will justify its features in columnar entries later in the paper.

In comparison with Fast Retransmission, Mobile TCP, and Freeze TCP, the differences/contributions of our TCP-M6 include/contain: For transport performance improvement for mobile Internet, first our TCP-M6 aims at specific/concrete mobility solutions for/to mobile Internet, namely Mobile IPv6, while the previous schemes do not associate with specific mobility solutions and we still need to study if these schemes can be directly applied to real mobile environments with some specific mobility solutions. Therefore, our TCP-M6 is based on mobile IPv6 environments, which makes our work have more practical significance. Secondly, we not only analyze the factors that contribute to the degradation of TCP performance in MIPv6, but also enhance the cooperation between the transport layer and network layer in our TCP-M6, which enable transport layer utilize the mobility-related information of network layer, thereby/so our TCP-M6 can perform better for TCP performance improvement. Thirdly, another important problem/issue of end-to-end transport delay deviation due to differences in/of network characteristics upon mobility/handover/motion can be observed in the paper and be solved by our TCP-M6. Fourthly, our TCP-M6 has the characteristic of adapting the packet size variation due

TABLE I

CHARACTERISTICS COMPARISON OF DIFFERENT PARADIGMS

Category	Network layer		Transport layer						
	HMIPv6	FMIPv6	Snoop	I-TCP	M-TCP	Fast Retransmission	Mobile TCP	Freeze TCP	Our TCP-M6
End-to-end semantics	√	√	√		√	√	√	√	√
Need third devices	√	√	√	√	√				
Handle link bit error			√	√	√				
Handle handover	√	√				√	√	√	√
MN as sender of TCP							√		√
Packet size variation									√
End-to-end delay deviation									√

to MIPv6 mobility protocol/management. Finally, although Mobile TCP [21] mentions/considers both scenarios that both endpoints may act as the sender of TCP, it does not handle them respectively, whereas our TCP-M6 in the paper can deal with the scenario, where the MN acts as the sender of TCP, beside the traditional scenario where the fixed node is the sender of TCP. We will justify its features in columnar entries later in the paper.

IV. PROBLEMS OF TCP IN MIPV6

As mentioned in the previous Section, the traditional TCP model originally does not take mobility issues into account. In this Section, we will investigate the adverse effects of mobility issues on TCP behaviours in MIPv6.

A. Mismatched CWND/SSTHRESH

In MIPv6, when the MN moves and switches between the access networks, MIPv6 may perform the following procedures: the movement detection, the configuration of CoA, and registration binding update. During these procedures, the communicating endpoints are not able to continue the communication between each other for some time, called handover latency. The handover latency depends on the indicated approaches [3], [14], [15]. Figure 3 illustrates the standard MIPv6 handover latency, which may induce transport delay, packet loss, and invoke the congestion control mechanism (see Section 2.2) in some subtle ways.

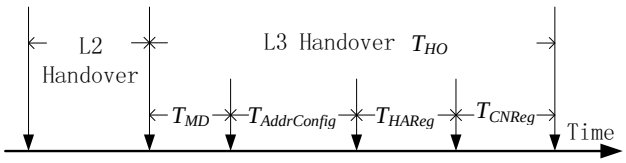

Fig. 3. The overall handover latency in MIPv6

Figure 4 visualizes the segment transmission from the CN to the MN during handover, where we assume that segment 1 is the last successful sent segment through the previous network. From this figure, we can see that when the MN moves out of its previous network at time t1 (i.e. the initiation of handover), because the TCP of CN is unaware of the occurrence, it may continue to send segments to the previous network of MN even if the MN has moved away (see figure 4 from segment 2 to segment k). Thus, any segments sent out by the CN from time t0 (i.e. half RTT before handover initiation) to time t2 (i.e. the termination of handover) will not reach the MN. This causes that if the timeout occurs before the handover termination, the slow start congestion control will be induced (see figure 4-a), which will drop CWND to the size of one segment; if the timeout does not take place during the handover latency, the MN may receive the new segment from the CN after handover if allowed by the current window size of the CN (see figure 4-b from segment k+1 to segment k+3), which will incur the CN to retransmit the lost packets and begin the fast retransmission congestion control because of the consecutive duplicate acknowledgements from the MN (see figure 4-b green segment 2).

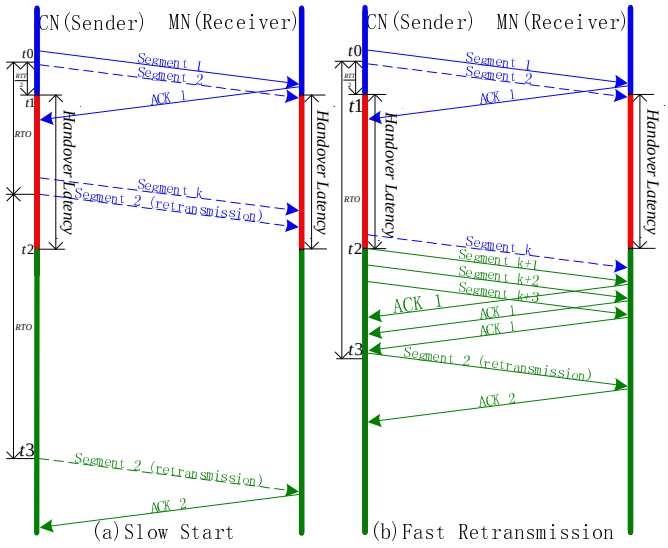

Fig. 4. Visualization of segment transmission from the CN to the MN during handover

Consider the segment transmission from the MN to the CN (see figure 5), then when the MN moves out of its previous network, all the segments sent but not yet acknowledged in the send window will not be acknowledged any more because of the unreachability of the acknowledgement messages, which are responded from the CN to its previous network of the MN (see figure 5 from segment 2 to segment k). If the lost ACKs do result in a retransmission timeout, the MN will reduce its CWND to one segment and return to the slow start stage (see figure 5-a); if the timeout does not take place during the handover latency, the CN may receive the new segment from the MN after handover termination if

allowed by the current window size of the MN (see figure 5-b segment k+m+1). Then the packets sent before t1 (see figure 5-b from segment 2 to segment k) may be confirmed after the new acknowledgment is delivered due to the cumulative feature of ACKs. However, segments sent during the time from t1 to t2 (see figure 5-b from segment k+1 to segment k+m) may be discarded by the interface of MN during the movement detection and CoA configuration procedures or by the HA during the stage of binding update procedure, which may likewise incur duplicated Acks retransmission or timeout retransmission (see figure 5-b).

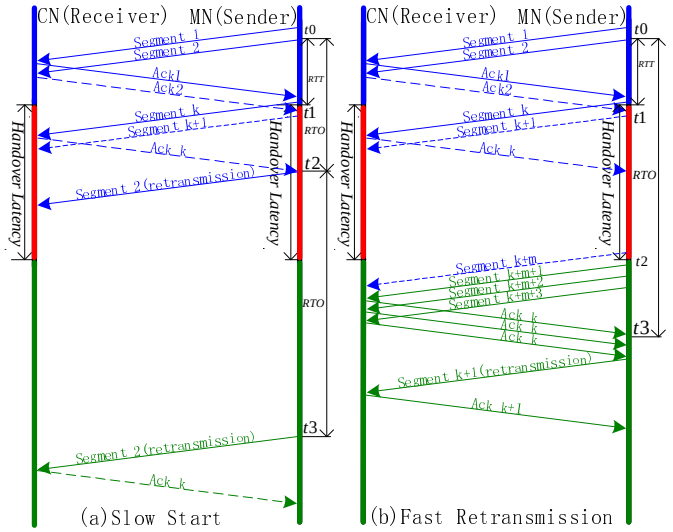

Fig. 5. Visualization of segment transmission from the MN to the CN during handover

In either scenario, successive lost packets (retransmission timeouts) result in a very small CWND and an equally small SSTHERSH. Figure 6 shows the behaviour of TCP CWND during MIPv6 handover.

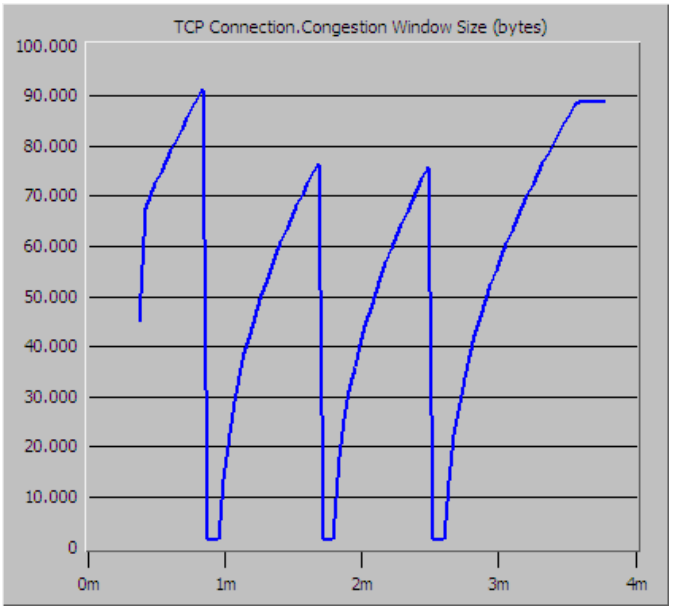

Fig. 6. Behaviour of TCP CWND in MIPv6

In addition, for the case of duplicate ACKs retransmission (see figure 4-b and 5-b), even if SSTHRESH is dropped to half of their previous values, which may also be stale for the new network. For example, when the new network has a lower capability (i.e. Bandwidth-Delay Product, BDP) than the previous network, the new setting may still be too high for the new network. The sender may inject more segments into the new network than would actually fit, causing the transmission queue at the bottleneck link's router to overflow. Conversely, if the new network is capable of carrying much more segments or more lightly loaded than the previous network, the value of SSTHRESH will become irrelevantly, and the increase of CWND is limited to one segment per RTT. It therefore will take multiple RTTs before reaching a reasonable throughput due to the slowness of additive increase for congestion avoidance.

B. Protean MSS

As described in section 2.2, for the maximum utilization of network, TCP may choose the MSS for the segment transmission. In traditional TCP, it assumes that the MSS is unaltered for the indicated active connection in fixed networks. Following this assumption, TCP may negotiate a MSS value while establishing a new connection. In practical implementation, the TCP MSS is determined as: MSS=MTU-IP_header-TCP_header, where MTU is the interface MTU of direct connected link of node; the TCP header takes up 20 bytes of data; the IPv6 header also uses 40 bytes.

In MIPv6, the IPv6 encapsulation header or Destination/Route extension headers are employed for mobile packet routing when the MN moves across different IP networks, which increases the packet size in the ongoing connection. Since the decision to add the mobility routing headers to the outgoing packet is performed dynamically at the network layer and is transparent to the TCP, this may result in the packet size exceeding the MTU, and as a result may incur the fragmentation/reassembly procedures at the network layer of sender.

Taking Ethernet with MTU 1500 Bytes for (as an) example, while establishing the connection over the MIPv6, the CN sends a SYN segment with MSS option that specifies a maximum segment size of 1440 bytes (Ethernet MSS = MTU (1500 bytes)-IPv6 header (40 bytes)-TCP header (20 bytes)). When the MN moves from a network to another network, the packets shall be transmitted between the MN and CN directly by appending the Type 2 Routing header (24 bytes) and/or Destination Option extension header (24 bytes) to the IPv6 packet in the RO mode, and the resulting IPv6 packet becomes 1524/1548 bytes (1440+40+20+24/48), which is greater than Ethernet MTU (1500 bytes). Similarly, in the BT mode, the ongoing packet sizes will extend to 1540 bytes for appending the IPv6 encapsulation header. Therefore, fragmentation/reassembly of the ongoing IPv6 datagram occurs, which aggravates the overhead.

We define the performance metric of overhead as following:

$$Overhead\_Ratio = \frac{Total\_Header\_Size}{Total\_Packet\_Size} \quad (1)$$

For previous example, each 1460 bytes of the original IP payload is fragmented into two fragments. Both the fragmented packets contain the fragment header and the mobility extension headers because the extension headers are the unfragmentable part. The first fragment is allocated the maximum IP payload size according to the fragmentation algorithm [37]. Then consider the scenario where the endpoints communicating with the RO mode and one of them is mobile and the other is stationary, the first fragment carries 1428 bytes IP payload computed using the following formula:

$$\begin{aligned} &First\_Fragment\_IP\_Payload \\ &= Ethernet\_MTU - IPv6\_Header \\ &- Routing\_Header - Fragment\_Header \\ &= 1500 - 40 - 24 - 8 = 1428 \end{aligned} \quad (2)$$

The second fragment includes the remaining payload of the original datagram with 32 bytes according to the following formula:

$$\begin{aligned} &Second\_Fragment\_IP\_Payload \\ &= Original\_IP\_Payload - IPv6\_Header \\ &- First\_Fragment\_IP\_Payload \\ &= 1460 - 1428 = 32 \end{aligned} \quad (3)$$

Therefore, the overhead ratio without fragmentation and the overhead ratio in MIPv6 for all segments are as follows respectively:

$$\begin{aligned} &Overhead\_Ratio \\ &= \frac{IPv6\_Header + TCP\_Size}{Ethernet\_MTU} = \frac{60}{1500} = 4\% \end{aligned} \quad (4)$$

$$\begin{aligned} &Overhead\_Ratio_{MIPv6} \\ &= \frac{Overhead\_Ratio_{fst.\_frag.} + Overhead\_Ratio_{sec.\_frag.}}{2} \\ &= [\frac{IPv6\_HD + TCP\_HD\_Rtg\_HD + Frag._HD}{Ethernet\_MTU} \\ &+ \frac{IPv6\_HD + Rtg\_HD + Frag.\_HD}{Or.\_TCP\_Packet - Fst.\_Frag.\_Payload}]/2 \\ &= \frac{\frac{92}{1500} + \frac{72}{104}}{2} = 37.9\% \end{aligned} \quad (5)$$

From the formula 4 and 5, we can see that after fragmentation, the overhead increases dramatically associated with MIPv6. This results in very inefficient use of bandwidth. Hence, the fragmentation needs to be avoided in order to improve the TCP performance in MIPv6.

C. Bogus RTT/RTO

Traditional TCP assumes that the deviation in RTT values would be small and constant. To avoid overreacting to a few very slow or fast acknowledgments and responding to consistent movement up or down in the RTT, the TCP retransmission timer specification [27] suggests estimating mean RTT via the low pass filter using a smoothing formula:

$$SRTT_{new} = (1 - \alpha) * SRTT_{old} + \alpha * RTT \quad (6)$$

Where $SRTT_{new}$ is the smoothed SRTT estimate, RTT is a round trip time measurement from the most recently acknowledged data segment, and α is a filter gain constant with a suggested value of 1/8 [38].

Considering the effect of load, Jacobson [38] introduces a factor of load RTT deviation, which is given as the formula [38]:

$$\begin{aligned} &RTTAVR_{new} = \\ &(1 - \beta) * RTTVAR_{old} + \beta* \mid SRTT_{old} - RTT \mid \end{aligned} \quad (7)$$

Where $RTTAVR_{new}$ is the round trip time deviation estimate, β is a filter gain constant with value of 1/4 as suggested in [38].

Once $SRTT_{new}$ and $RTTAVR_{new}$ estimates are updated, the retransmission timeout interval RTO for the next packet is set as the following formula:

$$RTO = SRTT_{new} + \max(G, K * RTTVAR_{new}) \quad (8)$$

Where G is a clock granularity, K is a constant parameter that is suggested as 4 [27].

In MIPv6, when the MN moves from one network to another network, the route between communicating endpoints will change and the deviation in RTT increase quickly. Figure 7 shows the deviation of RTT. However, as we see from above, the factors of α and β are low values, which slows down the quick TCP reaction to more frequent changes in the RTT. Therefore, the traditional TCP does not adapt the $SRTT$ and $RTTVAR$ to estimate the RTO during MIPv6 handover, which may violate this assumption and cause TCP to make the congestion control decision based on invalid variable values.

Besides, in the TCP retransmission mechanism, to eliminate the problem of acknowledgment ambiguity, Karn's algorithm [28] does not use the measured RTT for any segments that are retransmitted in the calculation of the overall retransmission timeout for the connection (see formula 8) and incorporates a timer backoff scheme for retransmitted segments. When a segment is retransmitted, it is backed off using a multiplier (typically 2, see formula 9, called exponential backoff) to give the retransmission more time to be received. The timer continues to be increased until a retransmission is successful, up to a certain maximum value.

$$RTO_{new} = 2 * RTO_{old} \quad (9)$$

In MIPv6, the successive timeouts will incur an artificial inflation of retransmission timer due to backing off RTO multiple times. As a result, the sender may waste long time to wait for a timeout before it can send new data continuatively (see figure 4 and figure 5 from t2 to t3). Figure 8 shows the behaviour of TCP RTO during MIPv6 handover.

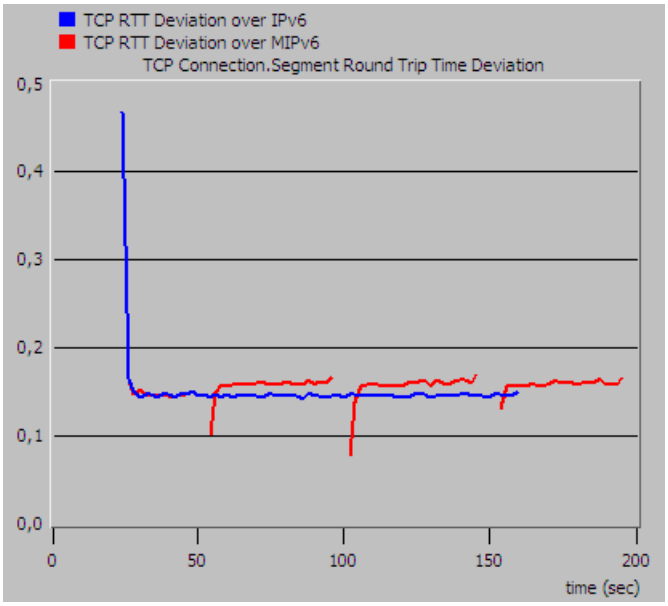

Fig. 7. TCP RTT deviation in MIPv6

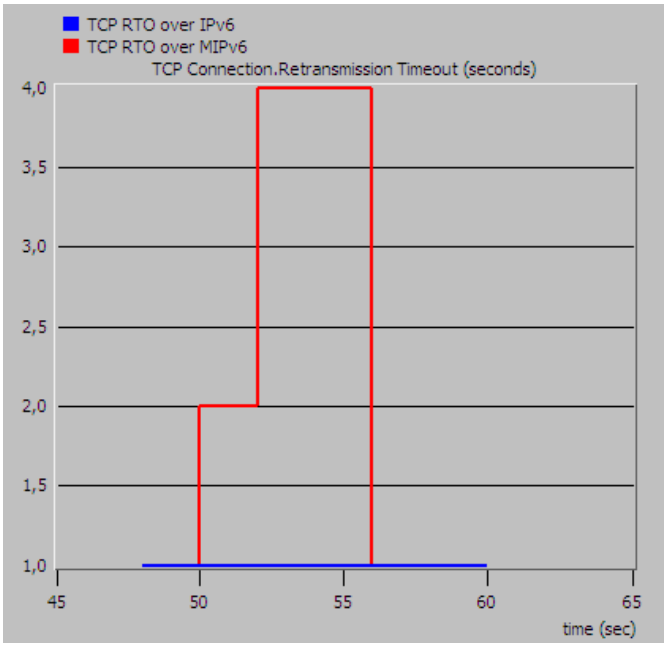

Fig. 8. Artificial inflation of RTO in MIPv6

From the illustrated bonus change of RTT/RTO, we can see that the RTT and RTO are not suitable for the new network due to movement.

V. TCP Mobility Control Mechanisms in MIPv6

Our analysis and simulation results demonstrate that, in highly dynamic network environments, it is difficult to maintain high transport performance as well as extended mobility functionality via a single-layer solution. We must enable TCP more actively to react to mobility issues in MIPv6. In this Section, we make modest changes to TCP and MIPv6 to enhance cooperation between them through inter-layer signalling interactions. Therefore, the mobility-aware TCP can response appropriately by tuning its behaviours according to the procedures of the mobility management protocol. The following subsections detail the two major aspects.

A. Enhancement of Inter-layer Interaction between TCP and MIPv6

To enhance the cooperation between TCP and MIPv6, MIPv6 must monitor the mobility-related procedures and is aware that subsequent procedures should be optimized by TCP. And TCP also requires a way to learn of these mobility-related events of MIPv6.

We define four types of mobility events in MIPv6: Initiation_of_Handover (IoH), L3_Mobility_Away (LMA), Termination_of_Handover (ToH), and Mobility_Routing (MR). At the MN, it can easily identify its handover initiation through the Neighbour Unreachability Detection mechanism [25], through which the MN can actively probe the current access router using unicast Neighbor Solicitation messages to verify if the forwarding route is still working. If the MN receives a solicited Neighbor Advertisement with the Solicited flag set, it confirms that the current access router is reachable. Otherwise it indicates the MN is moving out of its current network, causing the event of IoH. While the MN is aware of the IoH, it needs to check if the MN returns the home from a foreign network, or is moving away from the home network. MIPv6 determines the movement modes by listening to the router advertisements and comparing the network prefix of the source address within this advertisement with the network prefix of its home network. If the network prefix of the source address within the router advertisement does not equal the network prefix of the home address or the CoA of MN, the MN is handing over to the new foreign network (i.e. the event of LMA). For the event of ToH, we can determine the event through the binding registration mechanism, through which the MN will receive the successful binding acknowledge when handover terminates. And we use the Binding Update List (BUL) to determine whether the datagram is routed through the RO mode or the BT mode (i.e. the event of MR). MIPv6 may signal TCP following the mobility-related events. The signalling may be delivered through shared memory between MIPv6 and TCP in the same endpoint [16], or the communications between MIPv6 and TCP might be implemented via Inter Process Communication (IPC) [39] or even direct function calls.

At the CN, an additional communication step is necessary to inform TCP of the events occurring at the other endpoint (i.e. MN) of the connection. For the notification of IoH, it requires the MN to predict the reasonable time (i.e. a warning period) before which an actual Neighbour Unreachability occurs, which is quite a critical issue to estimate accurately in practice. Therefore, in our scheme, after notification of the completion of handover as described above, the MN forwards the signal of other events except the IoH over the new network to the CN. The signalling travels from the MN to the CN through normal IP route and can take either of two modes: It can be extensions of TCP, like explicit notification mechanism by setting a Explicit Mobility Notification (EMN) flag [40]

or an Explicit Handover Notification bit [41] in a specially marked TCP acknowledgement segment, or using a special option in TCP [42]. This mode simplifies some of the inter-layer issues at the CN. However, it has several drawbacks: the mobility notification may have to be redone for other transport protocols, it becomes difficult to share messages between different transport protocols, and the connection-oriented state maintained by TCP may not easily extend to save events for long periods. In addition, the inter-layer interaction of the MN is still inevitable. Another mechanism of the interaction between the MN and CN is that the IP layer makes use of the MIPv6 binding registration mechanism to transfer mobility-related events of the MN rather than using TCP extensions for this purpose since it will not be necessary for both facilities to be used at the same time, causing their functionality overlaps to some extent. In our scheme, we use the second mechanism through extending the Binding Update message with LMA/ToH/MR bits to indicate that the IP layer process of the CN receives mobility-related events, which then can be transferred to TCP through inter-layer interaction mechanisms.

As soon as TCP is aware of the mobility-related events of MN, it can react to mobility issues of MN by tuning the state parameters. The adjustments of state parameters are derived from the mechanisms described in the following subsection.

B. TCP Activities to Mobility in MIPv6

When the TCP is aware of the mobility-related behaviours of MIPv6, it invokes the mobility controls as the following mechanisms:

- Fast Recovery: Ignore the outstanding segments in the send window from before handover initiation. These segments or their ACKs, which are sent before handover initiation, are in flight. Although the RTO of these segments may not take place, they in fact can not reach the receiver any more. Therefore, these segments should be retransmitted immediately.
- Congestion Avoidance: Prevent the invocation of congestion control while the MN moving. Since it is common that the communicating endpoints change their route while communicating with each other, it is not suitable that each time route variation occurs, the window size starts from slow start algorithm. So we need a way to optimize CWND/SSTHRESH values by quickly probing the route's available capacity. The optimal congestion window size can be determined based on bandwidth estimation [43], [44]. And updating the congestion window is based solely on ACKs for data sent through the new network.
- Timing Pause and Reinitialization: Reset the bogus timing parameters. It is obvious that the RTT fluctuates significantly in mobile environments. Traditional TCP computes the smooth RTT by using previous values. And RTO depends on the smooth RTT. In principle, the retransmission timeout state for new network has the same requirements as that of a new connection. In order to avoid timeout mistakenly and artificial inflation of pause, the timing parameters RTT/RTO should be reinitiated to the value of RTT acquired from the new network.
- Adaptive segment size tuning: choose an optimal segment size over MIPv6 by tuning the MSS/PMTU dynamically, since it is observed that the optimal segment size depends on MSS/PMTU.

As we place above special mechanisms for TCP, the following subsections describe the mobility controls intended to identify these mechanisms supported by different type of MIPv6 entities.

1) Activities for Mobile Node: In the scenario, where the MN is sending data to the CN, When the MIPv6 of the MN determines that the current access router is no longer reachable through the Neighbour Unreachability Detection mechanism, it notifies TCP of the event of IoH. TCP is then prevented from going on sending segments to the IP layer and the counter of RTO is suspended immediately. When the MN receives a packet of valid Binding Acknowledgement, which indicates handover completion, it notifies TCP of the event of ToH. Once TCP receives the signal of ToH, it tunes the MSS and resumes the RTO. In [45], Pramil et al. propose a TCP payload adjustment method, through which the TCP asks the IP layer for the value of MIPv6 routing header each time it creates a new segment. Then it readjusts the TCP segment directly, but this could be inefficient because TCP,which follows the congestion control, typically calculates and caches several other state parameters derived from the size of TCP segment. In addition, the adaptive MSS announcement as described in [45] takes care of fragmentation at the two endpoints of a TCP connection, but it does not handle the case, where there is a smaller MTU link in the middle between these two endpoints. We therefore develop an asynchronous notification mechanism to indicate the changes of MIPv6 extension headers and PMTU. Each time the MN moves to a new network, these parameters may be tuned, depending on the modes of movement: If the MN moves away from the home network to a foreign network, TCP performs path MTU update and changes the MSS through new PMTU and subtracting the appropriate values of the extension headers; if the MN returns the home, TCP performs path MTU update and changes the MSS through new PMTU. After changing the MSS during a connection, the segment size may not be a sub-multiple of the send window. Because TCP performance can be reduced if the send window size is not an exact multiple of the segment size in use [46], we need to revise the send window size as well when the MSS value is tuned. We set the send window size to the greatest multiple of the new MSS. Finally, TCP may resume transmission if allowed by the windows size. If the usable window size is zero, the window size is increased by one segment, and then TCP goes on transmitting segments to the CN.

2) Activities for Correspondent Node: In the scenario, where the CN is sending data to the MN, because the reasonable predication of warning period for an impending handover is not practical since the measurement of RTT is often very coarsely, the motion of MN will be notified together with the binding registration messages along with the signal of ToH. As soon as TCP at the CN receives the signal of ToH, it responses

as follows:

At first, it tunes the MSS to be that at the MN. It also clears the SRTT and RTTVAR and reinitiates the RTO taken with the following formulas [27].

$$SRTT = R \tag{10}$$

$$RTTVAR = R/2 \tag{11}$$

Where, the R is the value RTT is observed over the new network.

Since the outstanding segments in the retransmission queue travel through the previous network, and will never reach the MN, these segments are retransmitted, but not invocate the congestion control mechanism. And the CN resets the CWND and SSTHRESH according to the Bandwidth Delay Product (BDP) of the new network, which represents a sender's TCP congestion window [47]. That is, the SSTHRESH is set equal to the available pipe size when connections are switched to the new network (i.e. $BWE * RTT_{new}$), the CWND is set to the SSTHRESH, and then congestion avoidance takes over, where BWE means the estimated bandwidth, and is measured through [43], [44], and the value RTT_{new} is measured over the new network. We call this mechanism as fast recovery.

VI. Simulations and Evaluations

In this Section, we evaluate the effect of the proposed mobility control mechanisms through simulations performed using OPNET simulator [48]. The simulation models are built by extending the mobility control mechanisms from the standard MIPv6 and TCP model. The simulation results demonstrate how the enhancement of TCP performance can be achieved in our scheme.

A. Model Modifications

Like the traditional TCP/IP model, in the OPNET model, the TCP and IP modules are not allowed to transmit control information except the data delivery. To enhance inter-signalling communication, we implemented a layered protocol interfacing between the TCP and IP modules through the Interface Control Information (ICI) mechanism. The ICI allows the IP module to notify mobility events and incurs corresponding mobility controls.

1) ICI Format Specification: An ICI format defines the structure of an ICI in terms of the attributes that it contains. An attribute's data type determines what sort of information it can store. In our ICI contents, we specify the following attribute names: handover_initiation, l3_mobility_away, handover_terimination, route_optimization, and pmtu etc.

Where, handover_initiation means the event that the MN determines that the current access router is no longer reachable through Neighbour Unreachability Detection mechanism; l3_mobility_away determines the MN's current position; handover_termination means the MN receives a packet with a valid Binding Acknowledgement; route_optimization determines whether the communication mechanism between the MN and the CN is the route optimization or the bidirectional tunnel; pmtu refers to the path MTU through the new network. The format specifications for ICI between IP and TCP are shown in figure 9.

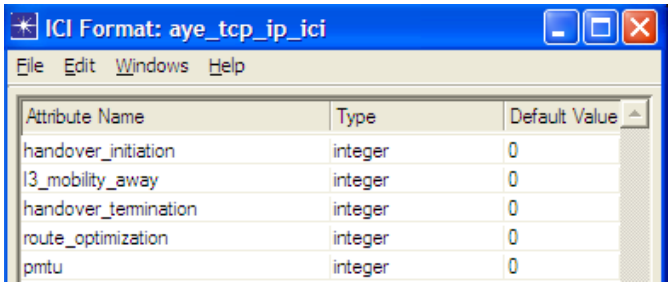

Fig. 9. ICI format specification

2) ICI Mechanics and its Implementation: The sequences of operation for the TCP and IP modules are as follows:

We create a new ICI at the IP module using the Kernel Procedure (KP) op_ici_create(), and supply the ICI format specified in previous Subsection. Then we associate the created ICI with the events of IoH and ToH respectively by calling the KP op_ici_install(). When the IP module is about to execute processes that will generate the events described above, and the ICI is to be associated with the corresponding events, it assigns correct values to the ICI's attributes by calling the KP op_ici_attr_set(). The following code for the indication and notification of IoH is implemented in the *away* state of the mipv6_mn process model:

```
node_objid = op_topo_parent (op_id_self ());
 tcp_mod_objid = op_id_from_name (node_objid, OPC_OBJ-
TYPE_PROC, "tcp");
 if (aye_handover_ici_ptr == OPC_NIL)
    {
    aye_handover_ici_ptr = op_ici_create ("aye_handover_ici_
   format");
    } endif
  op_ici_attr_set (aye_handover_ici_ptr, "aye_handover_initia-
   tion", 1);
 op_ima_obj_attr_get (compound_attr_objid, "Route Optimiza
   tion", &aye_opt_flag);
 if (aye_opt_flag)
    {
    op_ici_attr_set (aye_handover_ici_ptr, "aye_route_optimiza-
   tion", 1);
    } endif
 if (!L3_HO_HOME)
    {
    op_ici_attr_set (aye_handover_ici_ptr, "aye_l3_mobility_
   away", 1);
    } endif
 op_ici_install (aye_handover_ici_ptr);
 op_intrpt_force_remote(AYE_MOBILITY_CONTROL_INTRPT_
CODE, tcp_mod_objid);
```

When the indicated events occur and result in an interrupt, the interrupted process in the TCP module obtains the ICI by calling the KP op_intrpt_ici(). TCP extracts information that it needs from the ICI by calling the KP op_ici_attr_get(),

and responds correspondingly. Figure 10 shows the extended state diagram of the TCP process model. In the figure, we create a new state of MOB_CTR and define the condition of MOBILITY_CONTROL, which is associated with mobility-related events. As soon as the condition is true, it will transit to the MOB_CTR state, and perform corresponding Enter Executions as described below.

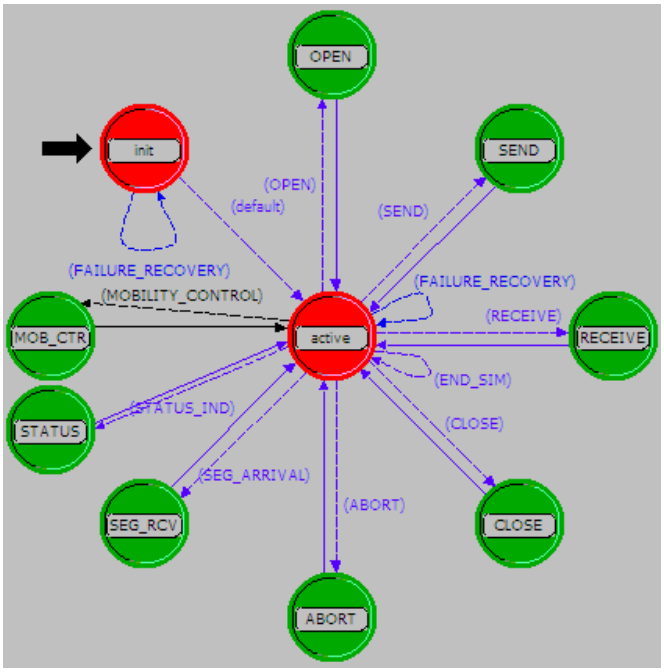

Fig. 10. The extended TCP process model

The following code is implemented in the extended MOB_CTR state of the TCP process model:

```
if (op_ici_attr_get (aye_ici_ptr, "handover_initiation", &tcp_
handover_initiation) == OPC_INITIATION_OF_HANDOVER)
  {
  current_time = op_sim_time ();
  avail_wnd = 0;
  } endif
if (op_ici_attr_get (aye_ici_ptr, "handover_termination", &tcp_
handover_termination) == OPC_TERMINATION_OF_HAND-
OVER)
  {
  if (op_ici_attr_get (aye_ici_ptr, "l3_mobility_away",
&tcp_l3_mobility_away) == OPC_L3_MOBILITY_AWAY=1)
    {
    if (op_ici_attr_get (aye_ici_ptr, "route_optimization",
&tcp_route_optimization) == OPC_ROUTE_OPTIMIZATION)
      {
      snd_mss=pmtu-IPV6C_DGRAM_HEADER_LEN_BYTES
      20-24;
      }
    else
      {
      snd_mss=pmtu-2*IPV6C_DGRAM_HEADER_LEN_BYTES
      20;
      } endif
  else
    {
    snd_mss=pmtu-IPV6C_DGRAM_HEADER_LEN_BYTES-
      20;
      }
    } endif
  retrans_rtt = measured_rtt;
  retrans_rtt_dev = measured_rtt/2;
  current_rto = retrans_rtt + rtt_dev_coef*retrans_rtt_dev;
  ssthresh = (bw*retrans_rtt)/snd_mss
  cwnd = ssthresh
  snd_wnd = int(snd_wnd/snd_mss)*snd_mss
  if (snd_wnd<=snd_nxt-snd_una)
    {
    snd_wnd = snd_wnd+snd_mss
    } endif
  next_timeout_time = next_timeout_time - (op_sim_time ()-
  current_time);
  avail_wnd = snd_una + total_wnd - snd_nxt;
  } endif
```

B. Experimental Set Up

To evaluate the effect of our scheme, the TCP performance test is carried out in our mobile IPv6 test environments, as illustrated in figure 11. The simulation network topology model is composed by the IP cloud model, Access Network (AN) model, Access Routers (ARs), and communicating endpoints (i.e. the MN and the CN) etc. The IP cloud model represents the Internet through which the IP traffic can be modelled. The access network model represents the access network technology with different features and capacity. Among them, the access network 1 refers to the home network modelled as 100Mbps rate with 0.01s delay and the access network 2 refers to a foreign network modelled as 10Mbps rate with 0.1s delay. AR represents the wireless access base station of access networks. And each AR consists of two interfaces, among which the wired interface is connected to the access network model through wired link and the wireless interface running IEEE802.11b with a coverage area of approximately 300 meters in radius provides wireless access for the MN. The HA (i.e. home AR) represents the home agent with home agent function, while the AR 2 (i.e. foreign AR) represents a foreign access router of access network 2. Each AR is positioned to be 250 meters apart with free space each other to ensure the overlapping distance.

The MN moves randomly within the coverage area of ARs following the mobility model (see the blue trajectory of MN 1 in figure 11) or deterministic path (see the green trajectory of MN 2 in figure 11) based on different simulation scenarios. For simulation operation, we define a FTP file transfer application through the Application object and Profiles object. The concrete application configuration will be depicted in the following subsections in detail.

C. Measurements and Results Analysis

In this Subsection, we measure the adaptive TCP behaviours for standard MIPv6 including the received sequence number, RTO, and CWND etc. To verify the correctness of our scheme. Besides, we also investigate the performance of overhead and throughput to validate the effect of our scheme.

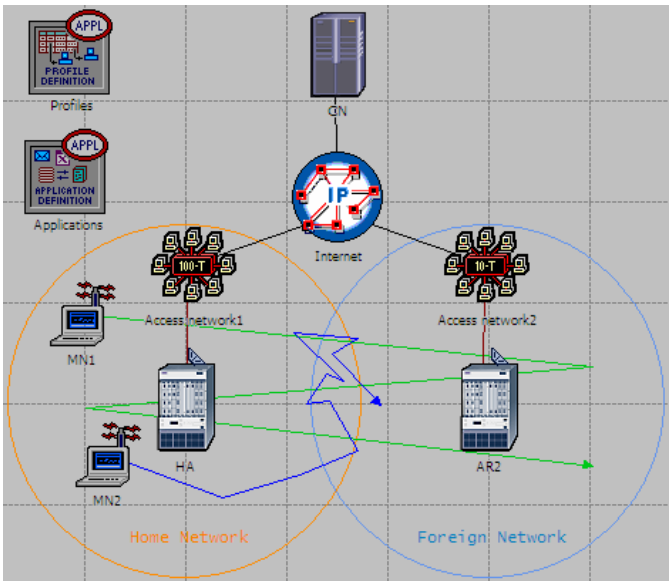

Fig. 11. The simulation network model

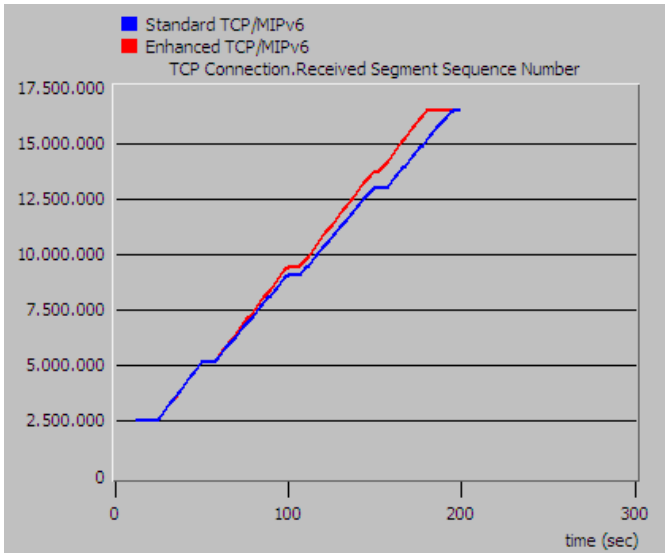

Fig. 12. The sequence number in TCP

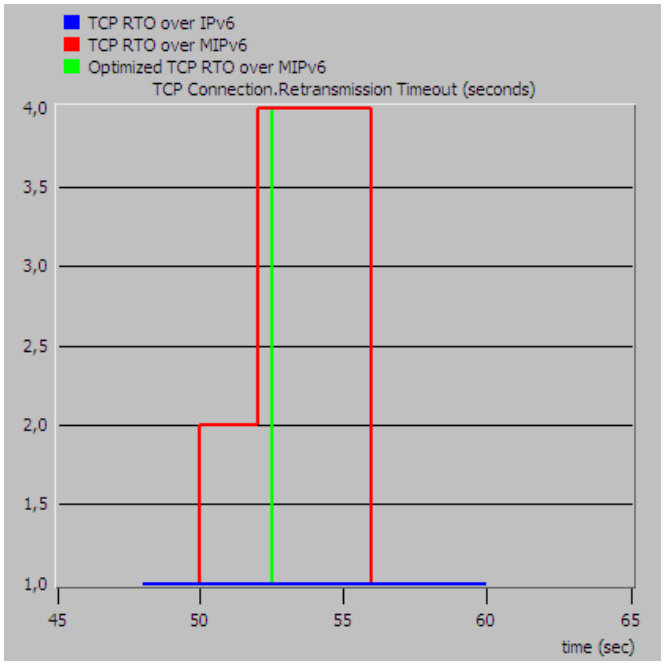

Fig. 13. Optimized TCP RTO in MIPv6

1) TCP parameter measurements and evaluation: First we measure the sequence number transmitted between the MN and the CN by running a FTP application. In our simulation, we use the standard TCP Reno model and set the CN as the FTP server and the MN as the FTP client. The FTP server was sending a file of 16Mbytes to the MN while the MN moves in the range of ARs, starting out at its home network, and switches to AR2 of foreign network in deterministic path with rate of 10m/s. This case allows for full control of the mobility and handover rate of the observed node. And we set the buffer size of ARs larger than the send window of TCP in order to avoiding packet loss due to buffer overflow of link layer, which eliminates the impact of link layer factor.

Figure 12 depicts the sequence number of the received TCP segments versus time for one of the experiments, from which the comparison of sequence number progression in a connection using our MIPv6-aware TCP and a connection using unmodified TCP can be observed. In the figure, the blue line is for unmodified TCP/MIPv6, and the red line is for the optimized TCP/MIPv6 with TCP mobility control. It can be seen from the figure that our proposed scheme recovers fast than the traditional TCP/MIPv6. This is because that we use the fast recovery mechanism, through which the TCP sender transmits data segments as fast as possible without waiting for an unnecessary timeout.

The retransmission timeout will be directly affected by the round trip time, i.e. SRTT and its deviation RTTVAR. In standard MIPv6, successive retransmission timeouts result in the artificial inflation of RTO. Figure 13 shows the optimized RTO with reinitialized SRTT and RTTVAR. From the graph, we can see that the RTO with reinitialized SRTT and RTTVAR is reduced significantly compared to the traditional TCP/MIPv6 as soon as the handover termination. When the TCP of CN is aware handover termination, it will reset the SRTT and RTTVAR, which leads to RTO recovery quickly.

We also trace the congestion window depicted in figure 14. In our experiment, we set the receive buffer size larger than the maximum size of CWND of TCP in order to avoiding the effect of size of the remote advertised window on send window, which eliminates the impact of the remote receive window factor on the send window.

Figure 14 plots the congestion window size with respect to the simulation time. From the figure we can see that at time 23 second when the FTP server of CN starts to transmit the file, the size of CWND starts to increase exponentially. At time 26 second, the CWND rises up to 65535 Bytes, which is the default value of SSTHRESH, and TCP enters congestion avoidance stage where CWND increases linearly. Whenever the connection is broken during the handover, the

standard TCP decreases the congestion windows size by half due to the fast retransmission caused by duplicated Acks, or one segments due to the timeout. However, our scheme does not invoke the slow start during mobility over MIPv6 (see figure 6 in Section 3). Each time the handover finishes, the TCP CWND is reset according to the BDP, which is usually higher than that of standard TCP. In this way, TCP prevents the invocation of any congestion control by estimating a suitable SSRESHTH and CWND and the congestion avoidance takes over.

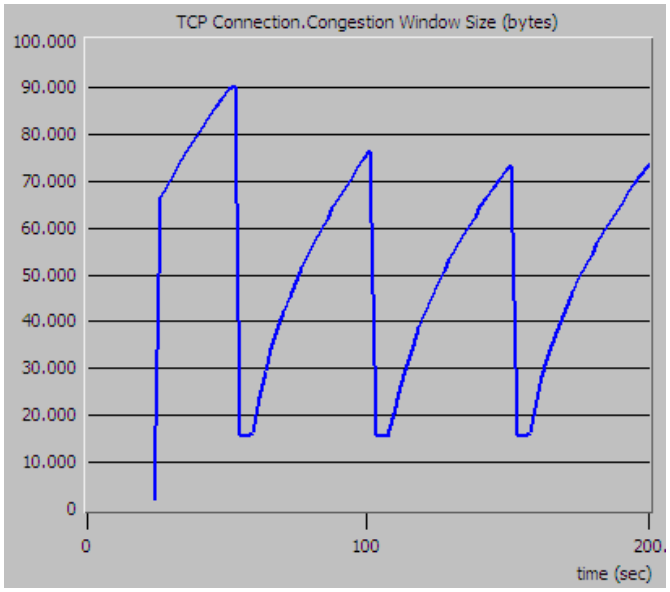

Fig. 14. The optimized TCP CWND in MIPv6

2) Performance measurements and evaluation: Overhead is a critical issue in wireless environments where there is constrained bandwidth resource. As we know in Section 2, MIPv6 employs the Tunnel header, Routing header and Destination Extension header for IP datagram routing control, which introduces the overhead. Moreover, IP fragmentation due to variation of segment size aggravates the problem of overhead. In order to evaluate the overhead performance over TCP-M6, we measure the overhead metric by simulating a TCP application.

Figure 15 shows the overhead ratio for TCP/MIPv6 with the MSS tuning mechanism. In this figure, the horizontal axis indicates the time in terms of second (sec) and the vertical axis indicates the traffic overhead ratio in terms of percent (%). Standard TCP/MIPv6 represents the traffic overhead ratio due to the addition of the IPv6 extension headers when routing data traffic using the MIPv6 RO mechanism. The TCP/MIPv6_With_MSS_Tuning represents the traffic overhead ratio when routing data traffic through the proposed MSS tuning mechanism.

It can be observed from the above figure that before time 50 second, the MN moves within access network1 (i.e. home network), and the overhead is about 4% with MSS 1440Bytes. At time 56 second, when the MN accesses to a foreign network (see figure 11, Access Network2), the average overhead ratio with traditional TCP rises up to about 38% drastically, whereas

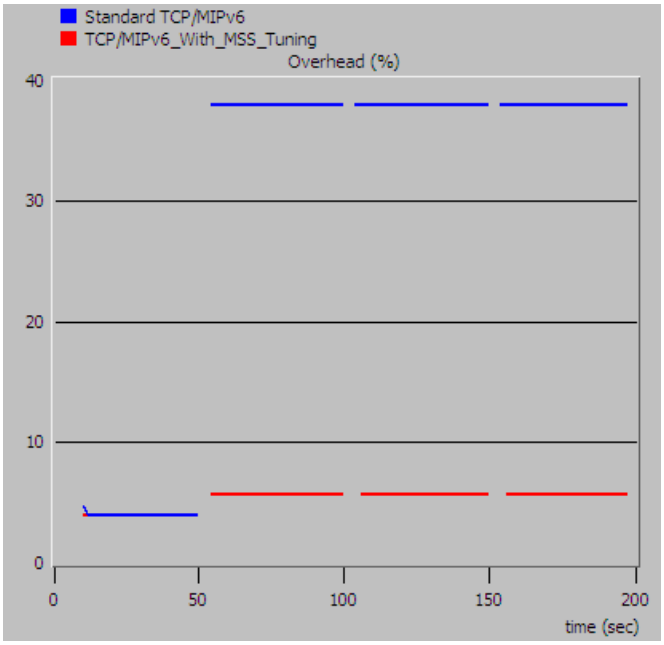

Fig. 15. The overhead ratio performance

the average overhead ratio with tuned MSS increases slightly up to 5.7%. It is clear that choosing an optimal segment size via our MSS tuning mechanism over a non-optimal segment size can improve the overhead performance by up to 30%.

We also investigate the significant improvements in the throughput over our TCP-M6. In the measurement, the MN was randomly moving across the wireless access area with various rates.

Figure 16 shows the throughput of TCP. In this figure, the horizontal axis indicates the time in which the endpoints are communicating with each other while the MN is roaming in terms of second (sec) and the vertical axis indicates the average throughput in terms of bits. The upper red curve is the throughput of TCP with mobility control, and the bottom blue curve shows the throughput with standard TCP. As illustrated in the figure, the throughput improves from 800000 bits/s to 850000 bits/s. In this way TCP improves the overall throughput through the mobility control mechanism because it differentiates the mobility with congestion and reacts correctly.

VII. Conclusions and Future Work

Like the network congestion, mobility is also an important issue that needs to be considered for TCP seriously in modern mobile environments. In particular, the mobile Internet over MIPv6 suffers from the packet loss, deviation of transport delay and variation of packet size etc, which are irrelevant to the network congestion. We have shown how handover procedures of MIPv6 cause the packet loss and inflation of RTO, presented how the deviation of transport delay causes the RTT stale, and quantified the variation of packet size aggravates the overhead. We have also identified the factors that contribute to the performance degradation.

To adapt TCP to mobility better as well as improve the transport performance in networks over MIPv6, we have also

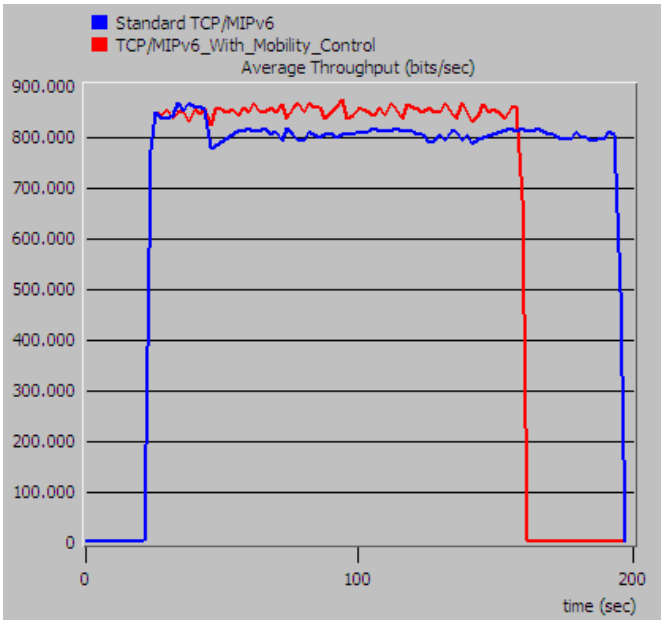

Fig. 16. The average throughput of TCP with mobility control

described mobility control mechanisms. Among them, the fast recovery and timing reinitialization mechanisms can eliminate the unnecessary waits for retransmission timeouts, and restore to a suitable network capability quickly; and the congestion avoidance mechanism prevents unnecessary invocation congestion control procedures after the MN moves to a new network, avoiding the degradation of throughput; and the adaptive segment size tuning mechanism adjusts the transmission segment size dynamically, optimizing the transport overhead performance.

Our work makes clear that the need for TCP to deal with mobility control and congestion control respectively. Our results can be used to adapt TCP to the mobile Internet over MIPv6. Simulation results show that mobility control is significantly more robust than traditional TCP in the presence of mobile networks over MIPv6.

While our approach does deal with many of the transport performance problems associated with MIPv6, we still have further work needed to be done. For example, we do not consider the effect of some wireless links due to the presence of bursty wireless channel error, which however is in existence in the real wireless networks. In addition, since the MIPv6 is still not the perfect mobility solution, it is necessary to investigate the integration of our TCP-M6 with the variants of MIPv6. In the future, we will study the effect of our approach in the real wireless networks with the presence of bursty wireless channel error, and we will integrate our TCP-M6 with the MIPv6 extensions (i.e. HMIPv6 and FMIPv6) for further expected performance improvement of TCP.

ACKNOWLEDGEMENTS

The use of OPNET Modeler in the research was facilitated through OPNET's university program.

REFERENCES

[1] R.H.Katz and E.A.Brewer, "The Case for Wireless Overlay Networks," Mobile Computing, Kluwer Academic Publishers, pp. 621–650, 1996.
[2] D.Le, X.Fu, and D.Hogrefe, "A Review of Mobility Support Paradigms for the Internet," IEEE Communications Surveys and Tutorials, IEEE, 2006.
[3] D.Johnson, C.Perkins, and J.Arkko, "Mobility Support in IPv6," RFC 3775, IETF, 2004. [Online]. Available: http://www.ietf.org/rfc/rfc3775.txt
[4] H.Soliman, "Mobile IPv6: Mobility in a Wireless Internet (Book)," Addison Wesley Longman Publishing Co., Inc, 2004.
[5] S. J. Vaughan-Nichols, "Mobile IPv6 and the Future of Wireless Internet Access," IEEE Computer, IEEE, pp. 18–20, 2003.
[6] M.Samad and R.Ishak, "Deployment of Wireless Mobile IPv6 in Malaysia," Proceedings of RF and Microwave Conference (RFM'04), IEEE, pp. 256–259, 2004.
[7] B. Aboba, "IAB Considerations for the Split of Identifiers and Locators," draft-iab-id-locsplit-00.txt, Internet Draft (work in progress), IAB, 2004.
[8] A. Fladenmuller and R. D. Silva, "The effect of mobile IP handoffs on the performance of TCP," Mobile Networks and Applications, ACM, pp. 131–135, May 1999.
[9] N.A.Fikouras, K. Malki, S.R.Cvetkovic, and C.Smythe, "Performance of TCP and UDP during mobile IP handoffs insingle-agent subnetworks," Proceedings of Wireless Communications and Networking Conference (WCNC'99), IEEE, pp. 1258–1262, 1999.
[10] S. Antoine, M. Wei, and A.H.Aghvami, "Impact of mobile IPv6 handover on the per-formance of TCP: an experimental testbed," ACM SIGMOBILE Mobile Computing and Communications Review, ACM, pp. 31–33, 2003.
[11] J. Hu and D. Phillips, "Simulation study of TCP performance over mobile IPv4 and mobile IPv6," Proceedings of the 5th International Conference on Enterprise Information Systems (ICEIS'03), pp. 224–231, 2003.
[12] S. Chang, J. Park, Y. Won, and M. Yang, "Performance Comparison of TCP Traffic over Mobile IPv4 and IPv6 Networks and a Mobile Network Deployment Approach," Proceedings of the Fifth International Conference on Computer and Information Technology (CIT'05), IEEE, pp. 469–473, 2005.
[13] J.Postel, "Transmission Control Protocol," RFC 793, IETF, 1981. [Online]. Available: http://www.ietf.org/rfc/rfc793.txt
[14] H.Soliman, C.Castelluccia, K. Malki, and L.Bellier, "Hierarchical Mobile IPv6 Mobility Management (HMIPv6)," RFC 4140, IETF, 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4140.txt
[15] R.Koodli, "Fast Handovers for Mobile IPv6," RFC 4068, IETF, 2005. [Online]. Available: http://www.ietf.org/rfc/rfc4068.txt
[16] R.Caceres and L.Iftode, "Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments," IEEE Journal on Selected Areas in Communications, IEEE, pp. 850–857, 1995.
[17] H.Balakrishnan, S.Seshan, and R.H.Katz, "Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks," Wireless Networks, ACM, pp. 469–481, 1995.
[18] H.Balakrishnan, V.N.Padmanabhan, S.Seshan, and R.H.Katz, "A Comparison of Mechanisms for Improving TCP performance over wireless links," Transactions on Networking, IEEE/ACM, pp. 256–269, 1996.
[19] A.Bakre and B.R.Badrinath, "I-TCP: Indirect TCP for mobile hosts," Proceeding of 15th International Conference on Distributed Computing Systems(ICDCS'95), IEEE, pp. 136–143, May 1995.
[20] K.Brown and S.Singh, "M-TCP: TCP for Mobile Cellular Networks," ACM SIGCOMM Computer Communication Review, ACM, pp. 19–43, May 1997.
[21] M.Stangel and V.Bhargavan, "Improving TCP performance in mobile computing environments," Proceedings of International Conference on Communications (ICC'98), IEEE, pp. 584–589, 1998.
[22] T.Goff, J.Moronski, D.S.Phatak, and V.Gupta, "Freeze-TCP: A true end-to-end TCP enhancement mechanism for mobile environments," Proceedings of Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'00), IEEE, pp. 1537–1545, 2000.
[23] H. ElAarag, "Improving TCP Performance over Mobile Networks," ACM Computing Surveys, ACM, pp. 357–374, 2002.
[24] R.Droms, J.Bound, and B.Volz, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)," RFC3315, IETF, 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3315.txt
[25] T.Narten, E.Nordmark, and W.Simpson, "Neighbor Discovery for IP Version 6 (IPv6)," RFC 2461, IETF, 1998. [Online]. Available: http://www.ietf.org/rfc/rfc2461.txt

[26] A.Conta and S.Deering, "Generic Packet Tunneling in IPv6 Specification," RFC 2473, IETF, 1998. [Online]. Available: http://www.ietf.org/rfc/rfc2473.txt
[27] V.Paxson and M.Allman, "Computing TCP's Retransmission Timer," RFC 2988, IETF, 2000. [Online]. Available: http://www.ietf.org/rfc/rfc2988.txt
[28] P.Karn and C.Partridge, "Improving Round-Trip Time Estimates in Reliable Transport Protocols," ACM Transactions on Computer Systems, ACM, pp. 364–373, 1991.
[29] M. V.Paxson and W.Stevens, "TCP Congestion Control," RFC 2581, IETF, 1999. [Online]. Available: http://www.ietf.org/rfc/rfc2581.txt
[30] R. Hsieh, A. Seneviratne, H.Soliman, and K.El-Malki, "Performance analysis on hierarchical Mobile IPv6 with fast-handoff over end-to-end TCP," Proceedings of Global Telecommunications Conference (GLOBCOM'2), IEEE, pp. 2488–2492, 2002.
[31] R. Hsieh and A. Seneviratne, "A comparison of mechanisms for improving mobile IP handoff latency for end-to-end TCP," Proceedings of the 9th annual international conference on Mobile computing and networking (MobiCom'03), ACM, pp. 29–41, 2003.
[32] S.Jaiswal and S.Paxson, "Simulation-based performance comparison of TCP-variants over Mobile IPv6-based mobility management schemes," Proceeding of 29th Annual IEEE International Conference on Local Computer Networks (LCN'04), IEEE, pp. 284–291, 2004.
[33] H.Huang and J.Cai, "Improving TCP performance during soft vertical handoff," Proceedings of 19th International Conference on Advanced Information Networking and Applications (AINA'05), IEEE, pp. 329–332, 2005.
[34] G. M. Costa and H. R.Sirisena, "Freeze TCP with Timestamps for Fast Packet Loss Recovery after after Disconnections," Proceedings of International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS'06), SCS, 2002.
[35] F.Hu and N. K.Sharma, "The Quantitative Analysis of TCP Congestion Control Algorithm in Third-Generation Cellular Networks Based on FSMC Loss Model and its Performance Enhancement," Proceedings of Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'02), IEEE, pp. 407–416, 2002.
[36] S. Bhandarkar, N. E. Sadry, A. L. N. Reddy, and N. H. Vaidya, "TCP-DCR: A Novel Protocol for Tolerating Wireless Channel Errors," Transactions on Mobile Computing, IEEE, pp. 517–529, 2005.
[37] S. Deering and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification," RFC 2460, IETF, 1998. [Online]. Available: http://www.ietf.org/rfc/rfc2460.txt
[38] V.Jacobson and M.J.Karels, "Congestion Avoidance and Control," ACM SIGCOMM Computer Communication Review, ACM, pp. 284–291, 1988.
[39] C.Papadopoulos and G.M.Parulkar, "Experimental evaluation of SUNOS IPC and TCP/IP protocol implementation," IEEE/ACM Transactions on Networking (TON), IEEE/ACM, pp. 199–216, 1993.
[40] A.Gurtov and J.Korhonen, "Effect of Vertical Handovers on Performance of TCP-Friendly Rate Control," ACM SIGMOBILE Mobile Computing and Communications Review, ACM, pp. 73–87, 2004.
[41] H.Izumikawa, I.Yamaguchi, and J.Katto, "An efficient TCP with explicit handover notification for mobile networks," Proceedings of Wireless Communications and Networking Conference (WCNC'04), IEEE, pp. 647–652, 2004.
[42] Y.Swami, K.Le, and W.Eddy, "Lightweight Mobility Detection and Response (LMDR) Algorithm for TCP," draft-swami-tcp-lmdr-06.txt, Internet Draft (work in progress), IETF, 2005.
[43] N.Parvez and L.Hossain, "Improving TCP performance in wired-wireless networks by using a novel adaptive bandwidth estimation mechanism," Proceedings of Global Telecommunications Conference (GLOBECOM'04), IEEE, pp. 2760–2764, 2004.
[44] R.Wang, G.pau, K.Yamada, M.Y.Sanadidi, and M.Gerla, "TCP Startup Performance in Large Bandwidth Delay Networks," Proceedings of Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM'04), IEEE, Hong Kong, pp. 796–805, 2004.
[45] A.D.Pramil, S.Antoine, and A.H.Aghvami, "Enhanced TCP performance over Mobile IPv6: innovative fragmentation avoidance and adaptive routing techniques," Proceedings of Consumer Communications and Networking Conference (CCNC'04), IEEE, pp. 175–180, 2004.
[46] J.Postel, "TCP maximum segment size and related topics," RFC 879, IETF, 1983. [Online]. Available: http://www.ietf.org/rfc/rfc1983.txt
[47] W. Stevens, "TCP/IP Illustrated, Volume 1: The Protocols, 1 ed (book)," Addison-Wesley Professional, 1994.
[48] OPNET Technologies, Inc., OPNET Modeler, 2006. [Online]. Available: http://www.opnet.com/products/modeler/home.html