Cross-layer binding update for TCP performance enhancement over Mobile IPv6 networks

R.-C. Wang and R.-S. Chang

Abstract: Mobility support in Internet Protocol version 6 (IPv6) is a de facto standard for allowing nodes to remain connected while moving around in the IPv6 networks. To provide better handoff latency and performance, numerous optimisation techniques were developed. Some of them are single-layer optimisation such as the Hierarchical Mobile IPv6 architecture, which aims at reducing the registration time in network layer. Others are cross-layer optimisation such as the Fast Handover mechanism, which tries to reduce the address resolution time in network layer based on the data-link layer detection. Most of these techniques are focused on data-link and network layer optimisation. The authors propose a cross-layer mechanism, which uses the standard binding update packet in Mobile IPv6 handoff to enhance the overall transmission control protocol performance in transmission. The theoretical analysis and simulations show promising performance enhancements.

1 Introduction

The Internet Protocol version 6 (IPv6) [1] is the next generation protocol of the current IPv4 [2]. Recently, with the rising demand for wireless and ubiquitous networking, there are strong requests for Internet to become ubiquitous and to allow anytime, anywhere, anything and anyone network access. In mid-2004, the Internet Engineering Task Force approved the mobility support in IPv6 (MIPv6) [3] as a standard track in request for comments 3775. One of the new approaches taken therein compared to traditional Mobile IPv4 (MIPv4) [4] is that MIPv6 performs route optimisation to minimise propagation latencies and packet overhead. In MIPv4, all packets are tunnelled through the home agent (HA) to the mobile node (MN) in communication. Although MIPv6 route optimisation allows mobile peers to communicate via a direct path [5], the reduction in propagation latencies comes at the cost of increased handoff delays. Depending on the implementation, the delay may be up to four round-trip time (RTT) at the network layer. It has a great impact for real-time applications [6, 7].

Besides, there are also delays for router discovery and address configuration in standard IPv6 protocol stack. Those delays are in fact in the order of 100–1000 ms [8]. Numerous optimisation methods have therefore been developed to simplify network layer handoff procedures and reduce the handoff delays, for example optimisation in IPv6 address configuration [9, 10], router discovery [11–13] and movement detection [14, 15].

From the users’ perspective, the overall performance of a communication will be affected by all the latencies inside the seven ISO open system interconnection (OSI) layers. Thus, optimisation techniques for any layer will always be useful. In data-link layer, optimisations such as the data-link layer attachment procedures [16, 17] or network-access authentication improvement [18, 19] will no doubt enhance the handoff performance.

Due to the physical nature of wireless media, transmission control protocol (TCP) performance degradation in wireless networks is a well-known problem. The research community has suggested various schemes to enhance TCP performance over wireless networks. Most researches can be classified into three categories: link-layer solution, split connection and TCP modification.

Link-layer solutions cache packets at the base station and perform local retransmissions across the wireless link. They can be TCP aware or unaware. Because the link layer has immediate knowledge of the dropped frames, it can respond faster than high-level protocols. Examples are ‘Snoop’ [20], ‘TULIP’ [21], ‘Delayed duplicate acknowledgments’ [22] and ‘Reliable TCP-aware link-layer retransmission for wireless networks’ [23]. Split connection schemes split the wired and wireless networks at the point where the two networks meet. Since the two networks are completely different, the base station keeps one TCP connection with the fixed host, while at the same time it uses another protocol designed to perform better over wireless links for the MN. Indirect-TCP [24], mobile-TCP (M-TCP) [25], wireless-TCP [26] and TCP over wireless networks using multiple acknowledgments [27] use this approach. However, they always need to modify the base station and/or the mobile host.

TCP modifications focus on making the sender aware of the wireless link error by handling segment losses caused by wireless link errors or losses due to network congestion differently. This approach requires the base station to send explicit congestion messages to the sender or a mechanism to detect the causes of loss at the sender. Examples are ‘Fast retransmission’ [28], ‘ECN’ [29], ‘TCP Westwood’ [30], ‘TCP-Jersey’ [31], ‘TCP Veno’ [32] and ‘JTCP’ [33].

All the mechanisms described above can enhance the TCP performance in an IP-based wireless network. Besides these single-layer approaches, cross-layer mechanisms are another rising research area. Until recently, there
are not many cross-layer optimisation mechanisms. A few studies reduce handoff delays with a combination of different optimisations in data-link layer and network layer [34–36]. Fast Handovers for Mobile IPv6 (FMIPv6) [37] is one of the well-known approaches in these cross-layer mechanisms. However, to the best of our knowledge, no mechanism has ever been considered for cross-layer performance optimisation above the network layer to date.

In this paper, we propose a novel cross-layer optimisation approach. The binding update (BU) packet in MIPv6 route optimisation (network layer) is used to improve the TCP (transport layer) performance. We make some modifications in the TCP slow-start and congestion control procedures. It can be appended to the traditional TCP protocol stack as an option, and works together with both classical and new TCP stack. The characteristic of its user transparency also makes it easy to be deployed in the current network. It can also co-exist with other optimisation mechanisms.

The rest of this paper is structured as follows. Section 2 provides an introduction to mobility support in IPv6. Section 3 describes our approach. Section 4 details the performance analysis and simulation results. Finally, the paper is concluded in Section 5.

2 Mobility support in IPv6

MIPv6 provides mobility support for IPv6. It defines the network layer handoff procedure when an MN roams to a new foreign network (or back to its home network). This begins with a change in the link-layer attachment, followed by router discovery, address configuration and finally MIPv6 registrations. Fig. 1 illustrates the entire handoff procedure.

To simplify the following discussion, the data-link layer latencies and internal processing delays in all nodes are ignored.

2.1 Movement detection and router discovery

Movement detection is commonly implemented by analysing the different prefixes advertised in router advertisement (RA) messages and by probing reachability of previous routers attached. When an MN receives new prefixes and finds that the previous prefixes in use are no longer advertised, it will start to move to a different network. Generic movement detection uses neighbour unreachability detection message to detect the reachability of previous router [38]. Since an MN may be covered in single or multiple networks, this can happen before router discovery process or even after address configuration.

According to the IPv6 neighbour discovery standard [38, 39], an MN learns its new local routers and prefixes during router discovery. After the data-link layer handoff, there are two methods for an MN to receive the necessary RA message. First, the MN may send a router solicitation message from its link-local address to obtain the solicited RA message. Because the solicitations should be sent from the unspecified address, it must be performed after the MNs unicast address becomes invalid. Secondly, the local router will send its RA by multicast periodically. When a multicast RA is received, an MN finishes its router discovery process. In most cases, the second one will be faster because the MN must verify the uniqueness of the link-local address before it sends the solicitation. Address verification involves substantial delays. So we only show the second method in Fig. 1.

2.2 Address configuration

Because the new neighbours may have duplicate link-local address, an MN must verify the uniqueness of its link-local address after router discovery. The IPv6 stateless address autoconfiguration [40] is typical of this address configuration and reverification. Based on the different data-link types, the MN chooses an interface identifier combined from the link-layer addresses, machine serial numbers, and so on [41–46]. For some privacy concerns [47, 48] and special cases [44], a random number may also be generated as the interface identifier. In most cases, the IEEE
global identifier EUI-64 [49] is used to transfer the interface identifier to the 64-bits length. It then sends a multicast listener discovery report message [50, 51] to subscribe to the new solicited-node multicast group. The MN then verifies whether the address is unique using the duplicate address detection protocol. A neighbour solicitation message is transmitted for the address. Without any neighbour advertisement message received within a timeout period, the interface identifier is confirmed as unique and can be used as part of the care-of-address (CoA) with prefix advised from the local router.

Another choice for address autoconfiguration is dynamic host configuration protocol version 6 (DHCPv6) [52]. An MN can request its unique address from a DHCPv6 server. However, this stateful configuration requires a valid link-local address first. The link-local address must be configured through stateless address autoconfiguration as described above. This will lead to an initial delay before DHCPv6 solicitation. We will not be using this method in this paper.

2.3 Mobile IPv6 registrations

There are two modes for communications between an MN and its corresponding node (CN). The first one is bidirectional tunnelling. Packets between MN and CN are tunneled through the HA. The second one is route optimisation, which requires the MN to register its CoA binding to the CN. Packets between MN and CN then can be routed directly. This is a better solution for the triangular routing problem and also a fundamental part of the MIPv6 protocol. Thus, we use it as the basic mechanism in this paper.

2.3.1 Home registration: After the roaming, an MN has to register the new CoA to its HA. The home registration consists of a BU and a binding acknowledgment message. The BU notifies the HA of the new CoA. The binding acknowledgment message indicates the success of binding. Because the MN and HA can have pre-shared credentials to bootstrap an IPSec security association, the messages can so be authenticated and encrypted.

2.3.2 Correspondent registrations: The correspondent registration is more complicated because an MN may not share authentication credentials with all CNs, and the global public-key infrastructure is not yet well constructed. For many security reasons [53], the correspondent registration is instead protected through a return routability procedure. Only with this assurance is the CN able to accept the BU from the MN. This is done by the following four messages shown in Fig. 1.

- Home test init/home test: For the home-address test, the MN sends a home test init message to CN, which is forwarded by the HA. The CN returns a randomised home keygen token to the home address using a home test message tunneled through HA to MN.
- Care-of-test init/care-of-test: The CoA test is a direct exchange between MN and CN. It consists of a care-of-test init message and a care-of-test message with a randomised care-of keygen token.

When the MN has received both the home and care-of-test messages, the return routability procedure is complete. The MN can now receive packets at the home address and CoA.

The MN then creates the binding management key (Kbm) for use in sending a verifiable BU to the CN. Finally, the correspondent registration is finished after sending a BU message to CN that conveys the new CoA, and an optional binding acknowledgment message may follow.

3 New cross-layer BU mechanism

In this section, we describe the proposed cross-layer optimisation procedure to improve the TCP performances during BU.

3.1 Definitions

We assume that the reader is familiar with the standard TCP congestion control algorithms [54, 55]: slow-start, congestion avoidance, fast retransmit and fast recovery. Table 1 illustrates the definitions of terms used in this paper.

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>cwnd</td>
<td>Congestion window: a TCP state variable that limits the amount of data a TCP can send.</td>
</tr>
<tr>
<td>FlightSize</td>
<td>Flight size: the amount of data that has been sent but not yet acknowledged.</td>
</tr>
<tr>
<td>G</td>
<td>Clock granularity: a fixed timer for scheduling the retransmission timeouts in TCP.</td>
</tr>
<tr>
<td>LW</td>
<td>Loss window: the size of the congestion window after a TCP sender detects loss using its retransmission timer.</td>
</tr>
<tr>
<td>RTO</td>
<td>Retransmission timeout: the duration of retransmission timer that is used for TCP to ensure data delivery in the absence of any feedback from the remote data receiver.</td>
</tr>
<tr>
<td>RTT</td>
<td>Round-trip time: the time required for a network communication to travel from the source to the destination and back.</td>
</tr>
<tr>
<td>RTTVAR</td>
<td>Round-trip time variation: a TCP state variable that is used for calculating RTO based on every measured RTT value, SRTT, and its previous state.</td>
</tr>
<tr>
<td>RW</td>
<td>Restart window: the size of the congestion window when a TCP restarts transmission after an idle period.</td>
</tr>
<tr>
<td>SRTT</td>
<td>Smoothed round-trip time: a TCP state variable that is used for calculating RTO based on every measured RTT value and its previous state.</td>
</tr>
<tr>
<td>ssthresh</td>
<td>Slow start threshold: a TCP state variable that is used to determine whether the slow start or congestion avoidance algorithm is used to control data transmission.</td>
</tr>
<tr>
<td>SMSS</td>
<td>Sender maximum segment size: the size of the largest segment that the sender can transmit. In IPv6 network, it is based on the MTU value which is measured from the path MTU discovery algorithm, excluding the length of TCP and IPv6 headers.</td>
</tr>
</tbody>
</table>
In the following discussion and simulations, we assume that a TCP link is connected between MN and CN. The bulk data were sent from CN to MN.

### 3.2 Problem of TCP in mobile environment

Since the seven layers model of OSI was published by ISO in 1978, the network layer and transport layer are treated separately. Network layer provides switching and routing technologies, creating logical paths for transmitting data from node to node. Routing and forwarding are functions of this layer, as well as addressing and internetworking. Transport layer provides transparent transfer of data between end systems and is responsible for error recovery and flow control. It ensures complete data transfer.

The success of mobile technologies brings a lot of new standards and research in network layer, such as MIPv4 and MIPv6. In a mobile environment, the packets on flight to previous access network will get lost during handoff. Different protocols in transport layer use different mechanisms to process these packet losses. For user datagram protocol (UDP) transmission, the packet losses may be acceptable because UDP is transaction oriented that delivery and duplicate protection are not guaranteed. On the other hand, for TCP transmission, the retransmission mechanism has to be used for recovering these lost packets.

However, if we analyze the details of TCP flow, the problems appear. The handoff procedure in data-link and network layer always lead to multiple and successive TCP packet losses. At the sender side, without receiving the acknowledgments from the receiver, the sender will treat it as network failure or congestion and start the congestion control process. Besides, if the handoff latency is larger than the RTT between sender and receiver, all the packets within this RTT period will be lost because they are all on flight. This is true in most cases because the handoff latency is still quite large today. It also means that the sender will not receive duplicate acknowledgments and will not start the fast recovery in congestion control. The sender must wait until retransmission timeout (RTO) and retry the transmission. Furthermore, upon a timeout, congestion window (cwnd) must be set to no more than the loss window (LW) which equals one full-sized segment. Therefore the sender uses the slow-start algorithm to increase the window from sender maximum segment size (SMSS) to the new value of slow-start threshold (ssthresh) after retransmitting the dropped segment. Then the congestion avoidance again takes over. According to the standard TCP congestion control, the small value of ssthresh will make the congestion avoidance start very quickly in (2). This will cause a bad performance in TCP transmission after handoff. Another performance issue is caused by RTO timer backoff algorithm in (3). TCP can do nothing while waiting for timeout. This is a waste of bandwidth. Trying to overcome these problems, we propose a cross-layer BU mechanism as follows.

### 3.3 Cross-layer BU mechanism

As described in Section 2.3, in the correspondent registration, a BU is used by an MN to notify a CN of its current binding. Compared with MIPv4, it is a fundamental part of the MIPv6 protocol. Although it is a network layer packet, we use it as a special transport layer signal for current TCP connections. We call it cross-layer BU (CLBU). The concept of CLBU is useful in two ways. First, when CN receives a BU from MN, it means that the MN is ready to continue the transmission. TCP need not wait until RTO timeout. Secondly, CN knows that the previous lost packets are caused by the mobile handoff, not by the network failure or congestion. The network is now ready as before handoff. It can do something to improve its efficiency. To simplify the discussion, we assume that the MN is roaming in homogeneous networks. Thus, we can assume the parameters (bandwidth, latency, etc.) in both previous access network and new access network are similar. We do the following changes in (1) and (3) to define the new TCP congestion control procedure. The new equation is shown in (4) and (5).

\[
\text{ssthresh} = \begin{cases} 
\frac{\text{FlightSize}}{2}, & \text{SMSS} \\
\frac{l_{cwnd}}{2}, & \text{SMSS} 
\end{cases} 
\]

\[
\text{cwnd} = \begin{cases} 
\text{ssthresh} & \text{if } cwnd < \text{ssthresh} \\
\text{ssthresh} & \text{if } cwnd > \text{ssthresh} 
\end{cases} 
\]

\[
\text{RTO} = \begin{cases} 
1 & \text{if } \text{RTO} < 1 \\
0 & \text{no CLBU received} \\
\text{SRTT} + \text{max}(G, K \times \text{RTTVAR}) & \text{with a subsequent} \\
\text{RTO} \times 2 & \text{RTT measurement} \\
\text{RTO} \times 2 & \text{when timer} \\
\text{RTO} \times 2 & \text{backoff happens} 
\end{cases} 
\]

To reduce the waiting before RTO in (5), we reset the RTO immediately when a CLBU is received. A CLBU informs the TCP protocol stack that the packet losses are not caused by network failure or congestion. The CN now knows the MN is ready for communication. So the retransmission can start immediately. We call it ‘early restart’. The ‘early restart’ is used to reduce the waste from meaningless waiting. The CLBU gives us a signal of ‘ready to restart transfer’. The interval between CLBU received and traditional RTO is the time the early restart can save.

Equation (4) is another improvement due to CLBU. We set up a new parameter named \(l_{cwnd}\), which records the last cwnd value before handoff occurs. The value represents...
a suitable stable state condition if the network has run for a period of time. In (4), the ssthresh is set to half of \( l_{\text{cwnd}} \), if a CLBU is received, similar to the standard procedure of congestion control in a single packet loss. We call it ‘ssthresh tuning’. The ‘ssthresh tuning’ is used to reduce the impact from multiple packets losses discussed in Section 3.2. Once a CLBU is received, no matter how many packets are lost, we treat it as a single packet loss. Because packet losses are due to a handoff, good TCP performance can be predicted after CLBU is processed. The reason for using half of \( l_{\text{cwnd}} \) is to keep the original spirit of congestion control.

To implement these mechanisms, a procedure should be added to MIPv6 protocol stack at end hosts that may play the role of CN. In the MIPv6 protocol stack handling the BU packet from MN, some lines of codes should be inserted to inform the TCP protocol stack to perform CLBU. In the TCP protocol stack, a new function should be implemented to set the state variables as shown in (4) and (5). Because it uses the same communication signals among MN, HA and CN, the intermediate nodes such as routers and access points need not be modified. Another advantage is that it can co-exist with traditional CNs. If an MN sends a BU to a CN which has not implemented CLBU function, the TCP will act normally – waits for RTO and then continues. On the other hand, if an MN sends a BU to a CN which has implemented CLBU function (with or without ssthresh tuning mechanism), the TCP can have better performance. So the deployment can be done incrementally. This is important in user transparency for migration. Besides, because all functions below network layer remain intact, there is no doubt it can co-work with almost all other data-link layer or network layer optimisations as mentioned in Section 1.

4 Analysis and simulations

4.1 Theoretical analysis

The benefit of (4) is analysed as follows. Assume the best cwnd value of a transmission is best_cwnd, which is the smallest cwnd value that can carry most data in one RTT, or the largest cwnd limited by the queue in the path. Also assume the total handoff delay is larger than the initial RTO after packet loss. The benefit of (4) can be shown as follows. Assume the total handoff delay \( T_{\text{handoff}} \) falls in the \( n \)th RTO backoff period. That is:

\[
\sum_{i=0}^{n-1} 2^i \times RTO_i < T_{\text{handoff}} \leq \sum_{i=0}^{n} 2^i \times RTO_i, \quad n \geq 1
\]

Here the RTO\(_i\) means the ‘initial’ RTO value when backoff starts. Our ‘early restart’ mechanism will carry more throughput \( T_{\text{handoff}} \) as:

\[
T_{\text{handoff}} = \left( \sum_{i=0}^{n} 2^i - T_{\text{handoff}} \right) \times \text{best_cwnd}
\]

Substituting (10) into (11), we have \( 0 \leq T_{\text{handoff}} < (2^n \times RTO_i \times \text{best_cwnd}/\text{RTT}) \). It shows the benefit of early restart.

For example, assume a general MIPv6 network with minimum 5 Mbps bandwidth in each link, and with the following parameters values: SMSS = 1280 bytes, RTO\(_i\) = 1 s, RTT = 100 ms and \( T_{\text{handoff}} = 4 \) s. The value of best_cwnd and \( n \) can be calculated as best_cwnd \( \approx \) 62.500 bytes \( \approx \) 49\*SMSS and \( n = 2 \). Assume the transmission is at its best stable state before handoff. It means \( l_{\text{cwnd}} \approx \text{best_cwnd} \). For the worst case, we assume \( l_{\text{cwnd}} = \text{best_cwnd} \). From the equations above, the following values can be calculated: \( \Delta t^* = 4.8 \) s, \( \text{Th}^* - \text{Th}^* = 815 \) 360 bytes and \( \text{Th}^* = 1875 \) 000 bytes. This means we can carry an extra \( (\text{Th}^* - \text{Th}^*) + \text{Th}^* \approx 2.6 \) Mb of TCP data with our CLBU mechanism in 4.8 s after BU message received.

4.2 Simulation results

The network simulator, ns-2 [56], is used for our simulations. The standard ns distribution has been modified and extended for our CLBU implementation. The goal of

\[
\text{Th}' = \sum_{i=1}^{\text{best_cwnd}} i \times \text{SMSS}
\]

and the total throughput \( \text{Th}' \) of our ssthresh tuning in this period is:

\[
\text{Th}' = \sum_{i=1}^{\text{best_cwnd}} 2^{(i-1)} \times \text{SMSS} + \sum_{j=(\text{max}(l_{\text{cwnd}}/2, 2) \times \text{SMSS})}^{\text{best_cwnd}} j \times \text{SMSS} + (\Delta t^* - \Delta t^*) \times \text{best_cwnd}
\]

Subtracting (8) from (9), we can find that even in the worst case of \( \Delta t^* - \Delta t^* \geq 0 \), the throughput of our mechanism cannot be worse because \( \text{Th}' - \text{Th}' \geq 0 \).

Next, the benefit of (5) can be shown as follows. Assume the total handoff delay \( T_{\text{handoff}} \) falls in the \( n \)th RTO backoff period. That is:

\[
\sum_{i=0}^{n-1} 2^i \times RTO_i < T_{\text{handoff}} \leq \sum_{i=0}^{n} 2^i \times RTO_i, \quad n \geq 1
\]

Here the RTO\(_i\) means the ‘initial’ RTO value when backoff starts. Our ‘early restart’ mechanism will carry more throughput \( T_{\text{handoff}} \) as:

\[
T_{\text{handoff}} = \left( \sum_{i=0}^{n} 2^i - T_{\text{handoff}} \right) \times \text{best_cwnd}
\]
our simulation is to examine the average throughput between MN and CN with or without the two mechanisms we proposed. Because our mechanisms can be adapted to all TCP versions, the classical TCP version, TCP Tahoe, is chosen as the default TCP for performing the standard TCP congestion control. The network topology used for the simulation is shown in Fig. 2. This topology is a simple version of a typical Mobile network and has been used extensively in [57–59] as an example for discussion. Because our scenario is based on classical MIPv6, all routers are regular without any special function such as mobility anchor point.

An ns TCP source agent is attached to the CN and an ns TCP link agent is attached to the MN. The MN is initially positioned inside the old access network and starts to move towards the new access network. To simplify the simulations and comparisons, the handoff is fixed to happen at 20 s after the simulation starts. This is to allow the TCP communication to stabilise, that is TCP is transferring data with a full window. Also, the network layer handoff latency is fixed at a conservative value of 5487 ms, which is inherited from the average result of [58]. An FTP session between the MN and CN is started 1 s after the simulation starts. The bulk FTP data traffic is from the CN to MN. A single simulation run is 60 s in duration. Based on the minimum MTU required by IPv6, the TCP packet size is set to 1240 bytes which make the full IP segments become 1280 bytes after 40 bytes IPv6 basic header attached.

In all the figures of simulation results, there are three lines or plots in comparison. The first one which named ‘TCP Tahoe’ illustrates the results of original TCP Tahoe version working in MIPv6 environment. The second one which named ‘CLBU’ means our early restart mechanism, CLBU, is used with TCP Tahoe, but without applying the ssthresh tuning function. The last one which named ‘CLBU with ST’ shows the outcome with both CLBU early restart and ssthresh tuning mechanisms.

The first simulation results are shown in Figs. 3 and 4. Fig. 3 shows the average throughput between MN and CN. When the handoff occurs at 20 s, the average throughput suddenly decreases because of the break down of TCP connection. The throughput will gradually increase after TCP transmission is resumed. In the simulation, from the viewpoint of average throughput, CLBU is 25.00 kbps better than TCP Tahoe and CLBU with ST is 27.78 kbps better than TCP Tahoe. Fig. 4 plots the relations between packet sequence number and its arrival time from CN to MN. In the figure, we can find that TCP Tahoe continues the TCP transmission at around 27.1 s. Both CLBU and CLBU with ST continue the TCP transmissions at around
25.6 s. Because there is no bandwidth competitor in the network, the improvement for ssthresh tuning is insignificant. After 60 s of simulation, the successfully transmitted packets are 5025 for TCP Tahoe, 5175 for CLBU, and 5188 for CLBU with ST.

We run another simulation with some background data flows to simulate a more realistic environment. Another two MN are put into the simulation scenario, one is in old access network and the other is in new access network. Each node is attached a bulk FTP data flow from the right most router in Fig. 2. This will make the network congested in both wireless access networks. The MN must compete for its bandwidth before and after handoff. The results are shown in Figs. 5 and 6. In Fig. 5, we can find that the average throughput becomes a self-similar wave form because of the congestion in the network. In the simulation, it shows that CLBU is 5.42 kbps better than TCP Tahoe and CLBU with ST is 9.17 kbps better than TCP Tahoe in average throughput. Fig. 6 is the same as Fig. 4 with background traffic. The TCP transmissions for all three mechanisms continue at the same time as in the previous simulation. We can see that the points are sparser because of the network congestion. The effect of ssthresh tuning is more significant here because it can help TCP to go back to the stable state more quickly. After 60 s of simulation, the successfully transmitted packets are 1487 for TCP Tahoe, 1518 for CLBU and 1542 for CLBU with ST.

Two more simulations are constructed for showing the condition of more frequent handoff. After the first handoff, we force the MN to roam back to the old access router 10 s later. The other settings and parameters are the same as before. The results of single TCP dataflow are illustrated in Figs. 7 and 8, while those with background data flows are presented in Figs. 9 and 10. In these figures, after the first handoff, the TCP continues as in the previous simulations. During the second handoff, TCP Tahoe continues at around 37.1 s. Both CLBU and CLBU with ST continue the TCP transmissions at around 35.6 s. Fig. 7 shows CLBU is 50.70 kbps better than TCP Tahoe and CLBU with ST is 54.93 kbps better than TCP Tahoe in average throughput at 60 s. Fig. 8 shows the successfully transmitted packets are 4311 for TCP Tahoe, 4606 for CLBU and 4608 for CLBU with ST at the end. Fig. 9 shows CLBU is 12.50 kbps better than TCP Tahoe and CLBU with ST is 18.75 Kbps better than TCP Tahoe in average throughput. Fig. 10 shows the successfully transmitted packets are 1256 for TCP Tahoe, 1326 for CLBU and 1363 for CLBU with ST after the simulation. All the simulation results prove the CLBU mechanisms can really enhance the overall TCP performance.
5 Conclusions

In this paper, we propose a novel cross-layer BU mechanism. The major changes include early restart and ssthresh tuning. They achieve global optimisation in TCP data transmission by using network layer signals. They can work together with all existing MIPv6 implementations.

Because only a few processes need to be modified in the CN without changing the MN and the HA, they are easy for deployment. Especially they can be deployed partially such that transparency in migration will not be a problem.

There are still lots of cross-layer mechanisms that can be considered to improve the overall network performance, such as the use of data-link layer detection or movement prediction to reduce the delay and jitter in application layer. Their effectiveness remains to be seen.

6 References