Resolving TCP Re-ordering Issues on a Multi-homed NEMO-based Network

Chien-Ming Chou and Kun-chan Lan
Department of Computer Science and Information Engineering
National Cheng Kung University
Tainan, Taiwan
Jensen0915@gmail.com, klan@csie.ncku.edu.tw

Chen-Yuan Li
Department of Computer Science and Information Engineering
National Taiwan University
Taipei, Taiwan
D99922035@ntu.edu.tw

Abstract—A multi-homed Network Mobility (NEMO) architecture has been proposed previously to improve the reliability and aggregate the bandwidth of Mobile Router (MR) by simultaneously using multiple wireless links. In this paper, we discuss some potential issues that could affect the performance of TCP traffic in a multi-homed architecture. In addition, we propose a protocol to improve the performance of sending a single TCP connection over multiple wireless links on a MR. Our protocol consists of three components. First, we use a reordering buffer to eliminate unnecessary retransmission caused by out-of-order packets. Second, we develop a Multi-homed Split Connection mechanism to enable the TCP connection to quickly adapt to the conditions of multiple links when packet loss occurs. Finally, a Scheduled Window-based Transmission Control algorithm is proposed to mitigate the negative effects of out-of-order packets in the case of using heterogeneous links in a multi-homed environment. We evaluate the performance of our protocol by comparing it with the Concurrent Multi-path Transfer SCTP (CMT-SCTP) and show that it can resolve several performance issues.

Keywords—component; multihoming; tcp; reordering; split connection; packet scheduling.

I. INTRODUCTION (HEADING 1)

The Internet Engineering Task Force (IETF) has standardized an extension of the Mobile IPv6 protocol called Network Mobility (NEMO) [6] to support the on-board communication [1], [2]. In addition, several studies have proposed the use of multiple wireless links simultaneously to satisfy the bandwidth demand of an on-board network. However, regular TCP are not designed for a multi-homed environment, and the performance will be seriously degraded when data from a single TCP connection is stripped and sent over multiple links. Although, some transport layer protocols, such as the Stream Control Transmission Protocol (SCTP) [7], have been proposed to support multi-homing. They require changes in the protocol stack of the end host, which is impractical for real-world deployment. Given that currently more than 70% of Internet traffic is still TCP, it is unlikely that people will replace TCP with these new protocols. In addition, many of these protocols suffer serious degradations in performance when packet loss or out-of-order packets occur [9]-[11].

II. RELATED WORK

Our work builds on prior work in multi-homing, traffic stripping, reordering buffer, split connection and multi-path TCP.

TCP Multi-Home Options [12] was proposed in the IETF Internet draft to enable multi-homing in TCP by modifying the TCP stack and appending the necessary options into the TCP header. Multiple Care-of Addresses Registration MCoA[13] is an extension to mobile IPv6 protocol and is compatible with NEMO. MCoA allows a mobile node to register multiple care-of addresses. The mobile node can therefore use multiple interfaces in foreign networks. The Binding Identification number (BID) is then used to distinguish between multiple bindings pertaining to the same home address. In prior work Several Internet drafts [14], [15] have described the definitions
and requirements for IPv4 and IPv6 multi-homing. To allow explicit policy definition, they proposed a few implementation suggestions for interface selection at the end host. In addition, they identified some interface selection problems and focused on the connection-level scheduling. A prior work [16] discussed several traffic stripping schemes and implemented a multi-homing agent that supports failover and intelligent load balancing using connection stripping. This process strips connections across multiple links, and all data belonging to the same connection would be sent to the same interface for the duration of that connection. Using connection stripping, a single connection can only utilize one link’s bandwidth at a time, even when multiple links are available. In this work, ETOM is based on packet stripping, which means that the traffic of a single connection can be transferred through multiple links concurrently in a multi-homed NEMO architecture. Note that although we exploit the same architecture as in NEMO, the focus of our work is on improving the performance of TCP traffic in such an environment while the aim of NEMO protocol is to manage the mobility issues of an on-board network.

In ETOM, the HA has a reordering buffer whose function is similar to that in EFL-MP (Ethernet with Flow Label for Multi-Path) [17]. EFL-MP exploits the reordering buffer to eliminate unnecessary retransmissions due to out-of-order packets when sending TCP traffic over the Ethernet. In addition to the use of a reordering buffer, ETOM also employs another two techniques, namely, split connections and scheduled window-based transmission control, to further improve the performance of TCP in a multi-homed environment. Several Split TCP [18], [19] mechanisms have been proposed to improve the performance of TCP over error-prone wireless links. In Split TCP, the TCP connection is split into a wired part and a wireless part, with different transmission control schemes conducted in different parts. In this work, we adopt the idea of split connection for the multi-homed environment, and integrate it into our scheduled window-based control scheme.

Several transport-layer protocols have been proposed to aggregate end-to-end traffic over multiple paths. Their common drawback is that they are not compatible with regular TCP and require the modification of end hosts. pTCP [9] consists of two components: Stripper connection Manager (SM) and TCP-virtual. SM is used to strip the raw data traffic of a connection into multiple paths at the end point, while TCP-virtual handles per-path functionalities, and processes isolated transmission control for multiple micro-flows. Dhananjay S. Phatak et al [20] proposed a mechanism to aggregate the bandwidth of multiple IP paths through multiple interfaces using IP-in-IP tunneling. A packet sending algorithm was introduced to fully utilize the available network resources. However, their work required changes of the network stack at the end hosts, which could be impractical for real-world deployment. Concurrent Multi-path Transfer SCTP (CMT-SCTP) [11] proposed the CMT algorithm to achieve efficient parallel transfer of SCTP. The CMT algorithm consists of the following parts: Preventing unnecessary fast retransmissions, avoiding reductions in congestion window (cwnd) updates, and curbing increases in ACK Traffic. Multi-path TCP [21] has been proposed as a routing protocol to dynamically route TCP traffic through multiple paths, using assistance from the router to retrieve the information needed to decide which path to route the traffic through. A data distribution scheme called Arrival-Time matching Load-Balancing (ATLB) [7] was proposed for multi-path TCP communication, with the aim of solving the problem of out-of-order delivery in a multi-path environment. It calculates the data arrival time for each path and then schedules the packets in order to achieve in-order data delivery at the receiver. Unlike ETOM, Multi-path TCP and ATLB were not designed with a multi-homed environment in mind, and do not aim to use multiple links simultaneously to improve TCP performance.

III. PERFORMANCE ISSUE

In this section, we discuss some potential issues that could affect the performance of TCP traffic in a multi-homed architecture.

A. Spurious Retransmission and Unnecessary Congestion Window Reduction

In a multi-homed environment, a connection is prone to have out-of-order packets which result in duplicate Acknowledgements (ACK) in TCP, which causes unnecessary drops in the sending rate at the source node and low utilization of multiple links [10]. These out-of-order packets introduce spurious retransmissions and unnecessary congestion window (cwnd) reductions, which could seriously degrade the performance of TCP traffic.

B. Inappropriate Congestion Window Update

Regular TCP does not maintain independent variables for each link, such as the cwnd value, measured round-trip time (RTT) and so on. The TCP source only maintains a single cwnd value for a connection, even when this connection is striped over multiple links. A single cwnd value can not properly reflect different conditions of multiple links.

C. Inappropriate Congestion Window Update Queuing Delay in the Reordering Buffer

Different wireless access technologies, such as 3.5G, WiMAX and so on, could have different bandwidths and transmission delays. Sending TCP traffic in a multi-homed environment with heterogeneous links could potentially result in out-of-order packets and degrade the performance of TCP, as described previously. In fact, even with the use of a reordering buffer and different cwnd values for each link, the out-of-packets can still incur a large queuing delay at the HA and significantly reduce the sending rate of the source node.

IV. ENHANCEMENTS FOR TCP ON A MULTI-HOMED MOBILE ROUTER

To overcome the performance issues one might encounter when developing a MR for a multi-homed environment, we propose a protocol called Enhancements for TCP On A Multi-homed Mobile Router (ETOM). ETOM consists of three components: a Reordering Buffer, Multi-homed Split TCP...
Connections and Scheduled Window-based Transmission Control, which are described in details below.

A. Reordering Buffer (RB)

A simple solution to avoid spurious retransmission and unnecessary congestion window reduction is to deploy a reordering buffer on the HA. Out-of-order packets are buffered at the HA until the missing packets are received, and packets are then sent out in order to the destination. Therefore, the destination should always receive ordered packets, unless loss occurs when packets pass through the links between the HA and the corresponding node (CN).

B. Multiple Independent Transmission Control using Multi-homed Split Connection (MITC-MSC)

Spurious retransmission and unnecessary congestion window reduction are not the only problems harming the performance of TCP in a multi-homed environment. As discussed previously, maintaining only a single cwnd value for a stripped connection can not reflect the condition of multiple links in a multi-homed environment, and it is necessary to have independent transmission control for each link. Different links could have different network characteristics, such as different bandwidths and delays. Maintaining only a single rtt or cwnd value for multiple links is intuitively unreasonable. Therefore, we exploit the idea of split TCP and maintain independent TCP parameters for each link used by the stripped connection. For every uplink, we split a TCP connection into two at the MR, including one part between the on-board device and the MR and the other part between the MR and the CN. The latter is further divided into multiple sub-connections based on the number of links used by the stripped connection. Similarly, the downlink traffic is split at the HA, and this is termed multi-homed split connections (MSC). Each sub-connection will have its own set of TCP parameters, including cwnd, rtt, and packet sequence number, for its transmission control.

Furthermore, in a multi-homed environment, it will be difficult to detect packet loss based on the continuity of the sequence numbers in the packet header given that the data is stripped over multiple paths. Since all packets traveling between the MR and HA are IP-in-IP encapsulated [8], we exploit the Total Length field in the inner IP header for error control by utilizing this field for the error control of each sub-connection by putting a path sequence number in the inner IP header.

C. Scheduled Window-based Transmission Control (SWTC)

Even with the implementation of the reordering buffer in the HA, a large number of out-of-order packets can still lead to a large queuing delay in the reordering buffer and harm the performance of TCP traffic. This situation could potentially be exacerbated when links are more heterogeneous in a multi-homed environment. In this section, we describe a scheduling algorithm that can be used to mitigate the extra queuing delay due to reordering. The basic idea is to schedule each packet sent out through the “fastest” path when the MR has a packet to send. To estimate the fastest path, we calculate the expected duration required from transmitting a data segment until receiving its corresponding ACK. The fastest path is defined as the path with the minimum expected delay. The delay \( D \) consists of two parts: the waiting time \( w \) from now until the packet can be injected into a particular link, and the round-trip network delay \( r \).

Considering the possibility of packet loss, we estimate the expected value of delay \( E(D) \) as in Equation 1. Suppose random variable \( D \) can take delay value \( d_{loss_j} \) with loss probability \( p_j \), and \( d_{no loss} \) with additional probability. Here we define \( loss_j \) as the case when the same packet is continuously lost \( j \) times (i.e. having \( j \) retransmissions), where \( d_{no loss} \) is the estimated delay for the case when no packet is lost while \( d_{loss_j} \) is the estimated delay when the packet has been continuously lost \( j \) times. In order to simplify our analysis, we assume the time for loss detection (e.g. through timeout or observing gap in the sequence number) is included in \( E(D) \). The difference between

\[
E(D) = d_{loss_1} \cdot p_{loss_1} + d_{loss_2} \cdot (p_{loss_1} \cdot p_{loss_2}) + \cdots + d_{loss_j} \cdot \prod_{j=1}^{\infty} p_{loss_j} + d_{no loss} \cdot \left(1 - \sum_{k=1}^{\infty} \prod_{j=1}^{k} p_{loss_j}\right) \tag{1}
\]

where \( d_{loss_j} \) and \( d_{loss_j+1} \) is mainly one extra network round-trip delay. Therefore, we can estimate \( E(D) \) for different cases as follows. Here \( d_{loss} = d_{loss_0} + d_{no loss} \).

\[
\begin{align*}
  d_{no loss} & = w_{no loss} + r_{no loss} \\
  d_{loss_1} & = w_{no loss} + r_{no loss} + r_{loss_1} \\
  d_{loss_j} & = w_{no loss} + r_{no loss} + r_{loss_1} + \cdots + r_{loss_j}
\end{align*}
\]

In the above equations, \( w_{no loss} \) refers to the waiting time before the scheduled packet can be injected into the link and \( r_{loss} \) refers to the round-trip delay when there is no packet loss. \( d_{loss_1} \) refers to the case when the packet is lost once. Therefore, it will take one extra round trip delay \( r_{loss_1} \) for retransmission. Similarly, when the packet is continuously lost \( j \) times, \( d_{loss_j} \) will take \( j \) retransmissions and need extra round trip delays \( (r_{loss_1} + \cdots + r_{loss_n}) \).

It is difficult to estimate the loss detection time when a packet is lost and needs to be retransmitted by the MR. When continuous packet loss occurs, the MR can not detect the loss until it observes a gap in the received ACK. For example, assuming the MR sent data segments 2, 5, 7 and 8 through the same path to the HA, if segments 2, 5, 7 are lost, the MR can not detect these until it receives the ACK of segment 8. In other words, when every consecutive data segment is lost, the MR can not detect the loss until the RTO timer expires. Therefore, given that \( r_{loss} \) is difficult to predict, we simplify our equation by assuming \( r_{loss_j} = r_{no loss} \). Note that the estimation of \( r \) is used to estimate the expected delay, \( E(D) \), which is then used to select a ‘fastest’ path when the MR needs to schedule a packet. Therefore, what we concern about here is a comparison of the relative \( D \) values between different paths, and not their absolute values. In other words, we argue that approximation of \( r \) will not affect the accuracy of our results (shown in the next section), as long as the approximation is consistent over all the paths for every packet. As a result, we can simplify our formula as follows. Here we denote \( w_{no loss} \) as \( W \) and \( r_{loss} \) as \( r \).
In addition, we assume that the loss probability always remains the same and denote it as \( p \).
\[
d_{\text{ave},\text{loss}} = w + r,
\]
\[
d_{\text{ave},\text{loss}} = w + 2r,
\]
\[
d_{\text{ave},\text{loss}} = w + (j + 1)r,
\]

After simplifying \( w, r, p \), we can obtain \( E(D) \) as a function of \( w, r, p \) and replace Equation 1 with:
\[
E(w, r, p) = (w + 2r) * p + (w + 3r) * p^2 + \cdots + (w + (k+1)r) * p^k + \sum_{k=1}^{\infty} \left(1 - \sum_{k=1}^{\infty} p^k\right)
\]
\[
= \sum_{k=1}^{\infty} ((w + (k+1)r) * p^k) + (w + r) * \left(1 - \sum_{k=1}^{\infty} p^k\right)
\]

Equation 2 can be further reduced into a closed form as shown in Equation 3, which can be used to estimate the expected delay when a packet is scheduled to be transmitted on path \( i \).
\[
E_j(w_i, r_i, p_i) = r_i * \left(\frac{1}{1 - p_i^2} - \frac{1}{1 - p_i} + 1\right) + w_i, \quad \forall \text{Path } i
\]

To estimate the expected delay \( E(D) \), we need to first estimate \( w \) and \( r \). \( w \) is the waiting time before the scheduled packet can be injected into the link while \( r \) is the network round-trip delay (i.e. the duration between when the scheduled packet is injected into the link and when the MR receives the corresponding ACK). Estimating \( w \) needs to consider two things: the time \( m_i \) to wait for all previous scheduled packets to be injected into the link and the time \( n_i \) to wait until the congestion window is open (i.e. when cwnd is larger than the amount of outstanding data), as shown in Equation 4.
\[
w_i = m_i + n_i, \quad \forall \text{Path } i
\]

In other words, \( m_i \) can be computed by estimating the transmission delay of all previous scheduled packets. The transmission delay can be obtained by dividing the available bandwidth \( B \) by the packet size \( d \). We adopt the bandwidth estimation function in TCP Westwood [5] to calculate \( B \) and assume all packets have the same size. Take Fig. 2 as an example, packet 1 is sent at \( t_1 \) and the MR receives its ACK at \( t_3 \). The ACK of packet 1 triggers the scheduled transmission of packet 2 and packet 3. In this case, the waiting time \( m_i \) for packet 2 is zero. On the other hand, given packet 3 needs to wait for packet 2 to be transmitted, the estimated \( m_i \) for packet 3 will be \( \Delta t \).

\( n_i \) is the estimated time to wait for when there is enough bandwidth to transmit the packet (i.e. cwnd > the amount of outstanding data). In other words, \( n_i \) aims to estimate the duration for enough ACKs to come back so that cwnd can be advanced to a level that is larger than the amount of outstanding data on path \( i \). Finally, we use the measured rtt to estimate the network round trip delay \( r_i \) for path \( i \). The round trip delay consists of the transmission delay (including the transmission delay of those packets that are already in the scheduled queue waiting to be transmitted, i.e. \( m_i \)), queuing delay and propagation delay. Given that \( m_i \) is already estimated when we calculate \( w_i \), as shown in Equation 4, we estimate the round-trip delay \( r_i' \) for data segment \( k \) on path \( i \) as:
\[
r_i' = r_{i} + w_i + \Delta t
\]

We further compute \( r_i' \) by averaging samples \( r_i'' \) to avoid large fluctuations of \( r_i \) between different data segments. Here we set \( \alpha \) to 0.875 as in most implementations of smoothed rtt.
\[
r_i' = \alpha * r_{i-1} + (1 - \alpha) * r_i''
\]

D. Home Agent ACK in Multi-Homed Split Connection

When out-of-order delivery occurs, ACKs are delayed because of packet reordering, which results in the queuing delay in the HA and the reduced sending rate at the sender. Scheduled window-based transmission control may not be able to completely eliminate the queuing delay in the HA because it uses only an approximation to predict the expected delay of a path. Therefore, we propose a simple idea called home agent ACK (hACK) to mitigate the effect of such queuing delay. The HA will immediately send a hACK to the MR when it receives an out-of-order packet. The MR treats the hACK just like a normal ACK and advances the cwnd accordingly. Note that hACK can work independently, with or without the previously-described scheduled window-based transmission control.

V. PERFORMANCE EVALUATION

We evaluate the performance of ETOM in NS-2. We run FTP application on the source node. In order to fairly compare with CMT-SCTP, which runs on the end hosts (while ETOM runs on the MR and the HA), in our simulations we combine the source node with the MR, and the HA with the destination. The bandwidth and delay of the wired links are set to 100Mbps and 1ms. We simulate WiMAX for 10ms frame duration and 3.5MHz channel bandwidth. The delay of the WiMAX links is set to 14ms based on a real-world measurement study we conducted previously [4]. One link is using 2% loss rate in the simulation.

In Fig. 3, the performance MITC-MSC degrades quickly as links become more heterogeneous. Heterogeneous links tend to lead to out-of-order delivery and queuing delays in the HA. Given that in MITC-MSC each path maintains an independent transmission control, as links become more heterogeneous, the queuing delay might become larger than the RTO of some paths. As a result, when their RTO timers expire, the MR will reduce the cwnd of those paths to one. To make things worse, paths which have smaller RTOs are more likely to suffer such reductions in cwnd, and these paths are typically the ‘faster’
VI. CONCLUSION AND FUTURE WORKS

Out-of-order delivery and packet loss could occur in a multi-homed environment, which will potentially seriously degrade the performance of TCP. In this work, we propose a protocol ETOM which, unlike SCTP, is transparent to the end hosts and does not require any modification of the end points, which makes it practical for deployment in the real world. We evaluate ETOM’s performance by comparing it against SCTP in simulations and show it can achieve good performance in a lossy and heterogeneous multi-homed environment. In our future work we plan to implement ETOM on the smartphone to evaluate its performance and identify its limitations in a real world environment.

ACKNOWLEDGMENT

This work was supported by the National Science Council, Taiwan (R.O.C.) under grants NSC 100-2218-E-002-004 and NSC 101-2221-E-006-232.

REFERENCES