A Transport Protocol for Future Wireless Internets

Deddy Chandra[1] *MIEEE* and Richard J. Harris[2] *SMIEEE*
[1]School of Electrical and Computer Engineering
RMIT University, Melbourne, Vic, Australia
[2]Institute of Information Sciences and Technology
Massey University, Palmerston North, New Zealand
deddy.chandra@rmit.edu.au , R.Harris@massey.ac.nz

***Abstract* -- The traditional assumptions made by TCP about the operation of wired networks are often found to be invalid for wireless networks. Standard TCP semantics such as end-to-end flow control, congestion control mechanisms and error recovery provide reliability in wired networks. However, wireless communication systems have different characteristics when compared to wired networks that include higher bit error rates, higher latency, limited bandwidth, multi-path fading of the signals and handoff. In this paper, we propose an enhancement to TCP that we shall call *E-TCP*, which improves upon conventional TCP when it is applied to the wireless environment. Our simulation results show significant improvements to TCP performance with respect to packet loss detection.**

***Index Terms*— TCP, wireless network, congestion loss, corruption loss, handoff loss, congestion control.**

I. INTRODUCTION

THE Internet and wireless networks are arguably two of the most important technical developments that have contributed significant benefits to people's lives over the recent past. The next step in the evolution of these two developments will be convergence of the two technologies to provide wider-area wireless Internet access [16].

The Transport Control Protocol/Internet Protocol (TCP/IP) [14][16] suite is the protocol suite that integrates a wide range of different physical networks into a global Internet. TCP was developed to perform well in traditional wired networks where it serves as a transport protocol that guarantees reliable data transmission.

However, the wireless medium has different properties from those upon which the design of traditional TCP was based. In particular, TCP assumes that network links rarely corrupt data [11] and this leads to the assumption by TCP that loss of data is a signal that congestion is taking place in the network. Thus, upon the loss of a data segment, standard TCP reduces its sending rate in an attempt to alleviate perceived congestion in the network. This approach is effective if data losses are truly caused by congestion in the network. However, since wireless media have higher bit error rates leading to significant data corruption losses, such events can cause TCP to unnecessarily reduce its transmission rate, yielding a significant degradation in performance. Therefore, in order to improve TCP performance, a mechanism is required to differentiate between the possible causes of packet loss. If this can be achieved then TCP would be able to take appropriate actions according to the nature of the underlying cause of the packet loss.

Generally, there are two different approaches to improve TCP performance in a wireless network. The first approach attempts to hide any non-congestion related losses from the TCP sender and requires no changes to the existing implementation of the TCP sender. The reason is that, since the problem is local, the problem should be solved locally. As a result, most of the losses seen by the sender are due to congestion [6]. The second approach attempts to let the TCP sender become aware of the existence of a wireless environment, and consequently determine that some packet losses may not be due to congestion.

Many schemes and mechanisms have been proposed in an attempt to improve TCP performance over wireless networks. But none of them allows TCP to distinguish the cause of packet loss. It is possible to classify the schemes into three basic categories: (i) end-to-end schemes [8][10], (ii) link-layer schemes [6] and, (iii) split connection schemes [1][2].

The objective in the present work is to improve TCP performance over wireless networks. Our method falls into the second general category mentioned above, which is to make the TCP sender aware of the existence of a wireless environment and the possibility of corrupted packet losses in the connection. We propose an acknowledgment scheme together with an agent on an intermediate node that is positioned between the wired and the wireless network. This scheme allows TCP to distinguish congestion from corruption loss. Additional functionalities are required at the base station and at the TCP sender. We discuss the implementation of our proposed scheme and demonstrate its effectiveness through simulation.

The remainder of this paper is organised as follows. Section II reviews some related work in order to compare the advantages and disadvantages of existing solutions. TCP and its mechanisms are briefly described in section III. Section IV describes the assumptions concerning the inter-network structure.

Our proposed solution is described in detail in section V. An extension to our proposed solution to overcome handoff losses will be presented in section VI. A simulation model and results will be discussed in section VII and finally, some conclusions are presented in section VIII.

II. Related Works

In this section we review some earlier solutions that have been published in the literature. As we shall show, most of the existing schemes do not provide any solutions for coping with handoff loss.

The Last-hop acknowledgment scheme (LHACK) was proposed in [13]. In this scheme, the base station/wireless router sends an "LHACK" ("FHACK", first-hop acknowledgment) to the stationary (mobile) source for each packet that it receives. For a connection from a fixed host to a mobile host, if "LHACK" is received at the source but the "DACK" (acknowledgment from the receiver) is missing, this infers a corruption and no congestion control mechanism is triggered. By contrast, missing both "LHACK" and "DACK" implies congestion, thus the congestion control mechanism is invoked. The drawbacks of this scheme are that two acknowledgments are sent for each message and this will result in an extra load on the return path. This scheme must rely on timeouts and fast retransmission to detect corrupted packets and it does not consider the effect of handoff for the mobile host. Significant throughput degradation occurs when multiple corruptions occur in a single window. Furthermore, for connection between two mobile hosts, the loss of "LHACK" on the wireless link to the source can be misinterpreted as network congestion.

Explicit Congestion Notification (ECN) has been discussed in [11]. The ECN scheme marks packets when it senses congestion in the router. The TCP receiver informs the sender (in a subsequent acknowledgment) about incipient congestion when a congestion marked packet has been received, which in turn triggers the congestion avoidance algorithm at the sender. The major drawback is that it requires support from both routers as well as the end hosts. This scheme introduces more processing overhead on routers and it is not feasible to do so considering there could be many routers along the path.

The Negative acknowledgment (NACK) scheme was proposed in [2]. This scheme uses the TCP cumulative acknowledgement packet with additional "NACK" in the option field to explicitly indicate which packet is received in error so that the retransmission can be initiated quickly. The drawbacks in this scheme involve its reliance on the reception of an ACK packet. If the source does not receive the ACK, it could lead to expiration of the retransmission timer. Furthermore, this scheme does not consider communication between a fixed host and a mobile host where data packets may be lost due to congestion and corruption.

Our proposed E-TCP scheme was inspired by the acknowledgement scheme known as the Explicit Loss Notification with Acknowledgment (ELN-ACK) scheme, which was proposed in [14]. Unfortunately, ELN-ACK only considers the improvement of TCP performance on a communication between a fixed sender and a mobile receiver. In contrast, their scheme does not apply to communication between the mobile sender and a fixed receiver or communication between two mobile hosts.

Caceres and Iftode [8] proposed a fast retransmission mechanism. They focussed on improving throughput and reducing interactive delay to an acceptable level by forcing TCP to retransmit segments as soon as handoff is completed, without waiting for a TCP retransmission timeout. This approach requires the transport protocol to differentiate between motion-related and congestion-related segment loss. The advantages of this approach are that it requires modification to the end systems and relies on no special support from the underlying network or intermediate routers. The main disadvantage of fast retransmission is that it only considers host mobility (handoff) but it does not take into account the error-prone characteristics of wireless links.

III. Background

A. Transmission Control Protocol (TCP)

TCP is well known as a connection-oriented transport protocol. It regulates the number of data packets that it sends by inflating and deflating a window. In order to do that TCP, on the sender side, uses the cumulative acknowledgment packets sent by the receiver. TCP has been designed with a congestion control mechanism to overcome the common problem of packet losses due to congestion in a network. The congestion control mechanism in the regular TCP (Tahoe) [12] implementation has three main parts:

a. Slow Start
b. Congestion Avoidance
c. Fast Retransmit

More details of the TCP protocol can be found in [3] and [12].

B. Handoff Process

When a mobile host is engaged in a communication, it is connected to an active base station via a radio link. If the mobile host moves to the new coverage area of another base station, the connection to the old base station is eventually disconnected, and a new

connection to the new base station should be established to continue the communication. This process is referred to as *handover* or *handoff*.

During the handoff process, communication with the mobile host would be interrupted. Any data transmission to or from the mobile host during the handoff phase might be lost and these types of losses are known as handoff loss.

Since handoff losses are not triggered by congestion in the network, it is unnecessary for TCP to invoke its congestion control mechanism. This argument is similar to the argument for packet losses due to corruption.

IV. INTERNETWORK STRUCTURE

We assume that the inter-network model consists of multiple networks that are joined together by computers attached to one or more networks which are known as routers. Data is delivered from a computer in one network to another computer in another network via a router. The data is forwarded from one intermediate node to another until it reaches its final destination.

As shown in Fig. 1, the inter-network may consist of a group of wired networks that form the core of a wired network and a wireless network. In general, we may assume that data are relatively corruption-free in the wired network. Thus, it is reasonable to assume that in the wired network, data might be lost due to buffer overflows, i.e. congestion. Typically, wireless networks are not used to serve as intermediate hops in the routing path between two networks. Two or more computers in the inter-network can establish a TCP session between them, regardless of whether they are mobile or fixed hosts.

We also assume that routers provide only data delivery; therefore they do not guarantee its delivery. Routers forward each data packet to the next computer or router and this machine may drop the data due to buffer overflow (congestion). Routers in wired networks are not aware of mobile hosts and must not be affected by any changes that support wireless communication. We assume that enhancing the functionality of wireless routers to the minimum level is acceptable.

Wireless routers are not responsible for the reliability of data transmission due to avoidance of processing overhead and violation of TCP semantics. Routers that are attached to wireless networks are aware that a wireless network is prone to corruption and are located at the edge of the inter-network.

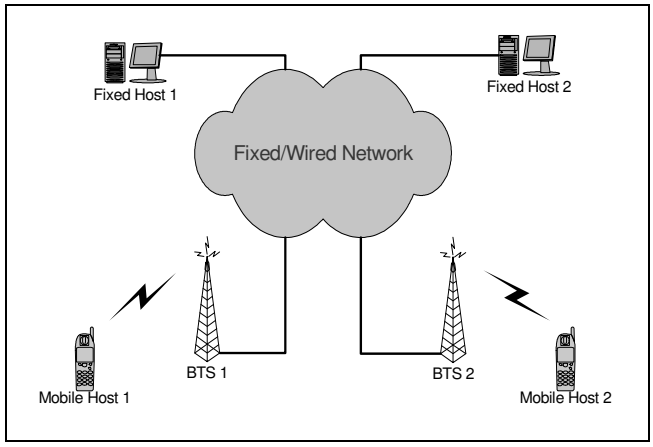

Fig. 1. Inter-networking Model

V. ENHANCED-TCP (E-TCP)

We propose a new enhanced TCP called E-TCP (Enhanced-TCP) that is able to differentiate packet losses due to corruption from congestion and takes appropriate action to overcome losses.

Further, we introduce the implementation of a transit agent (TA) at the base station[1]. A transit agent explicitly acknowledges E-TCP for the "transit history" of every packet that has transited at the base station. Thus, this acknowledgement is an indication for E-TCP to take appropriate recovery action over packet losses.

This scheme needs modifications to the structure of the acknowledgment packets and the software part of the transport protocol of the base station and the sending host. Many proposed schemes only considered communication between a fixed host and a mobile host, but our proposed scheme allows for communication between two mobile hosts as well (see Table 1).

Source Port											Destination Port
Sequence Number											
Acknowledgment Number											
Data Offset	Reserved			Fb	Sb	URG	ACK	PSH	RST	SYN	FIN	Window
Checksum											Urgent Pointer
Options											Padding
Data											

Fig. 2. Fb and Sb bits in the header of TCP segment

[1] A node that is located between the wired and wireless networks. For simplicity of understanding, we assumed here that a base station is the gateway node.

TABLE 1
CROSS-COMMUNICATION BETWEEN HOSTS

xmt \ rcv	Fixed Host	Mobile Host
Fixed Host	TCP	E-TCP
Mobile Host	E-TCP	E-TCP

A. *New Acknowledgment Packet*

We introduce a new form of acknowledgment packet that will be referred to as $ACK_{C\text{-}CLN}$ (congestion-corruption loss notification). This ACK packet requires up to two additional bits in the TCP header to store information set by the transit agent. Two of the six reserved bits in the TCP header must be allocated for this purpose (see Fig. 2).

When communication occurs between a fixed host and a mobile host, data transits only one base station. Thus, only one bit is used. If communication occurs between two mobile hosts this would typically involve two base stations (Fig. 1), two bits are used instead. For simplicity, we have named each bit in the $ACK_{C\text{-}CLN}$ packet. The first bit will be called "Fb" (First bit) while the second bit will be called "Sb" (Second bit). The first transit agent that receives an $ACK_{C\text{-}CLN}$ packet sets the Fb bit and the second transit agent that receives the $ACK_{C\text{-}CLN}$ sets the Sb bit.

B. *Transit Agent at the Base Station*

The proposed agent must be permanently allocated at the base station and it is only required to perform a simple task. The agent is required to record the sequence number of every packet that has transited through the base station before they get forwarded to the receiver. The flowchart for data processing by the transit agent is shown in Fig. 3. The agent also acknowledges the sender by providing the transit history of the expected packet through the use of $ACK_{C\text{-}CLN}$.

The following is the data processing algorithm performed by the transit agent:

```
if (data packet arrives) then
   pkt_seqno = get(SEQ_NO);
       pkt_transit[pkt_seqno]=TRUE;
end if
```

Note: initially the value of pkt_transit[pkt_id]=FALSE. Where FALSE=0 and TRUE=1.

Upon receiving of an $ACK_{C\text{-}CLN}$ packet from the receiver, the agent checks its records and sets a value for $ACK_{C\text{-}CLN}$ as follows: A "0" in the $ACK_{C\text{-}CLN}$ indicates the expected sequence number has never transited the base station before and a "1" indicates that the expected sequence number has transited the base station.

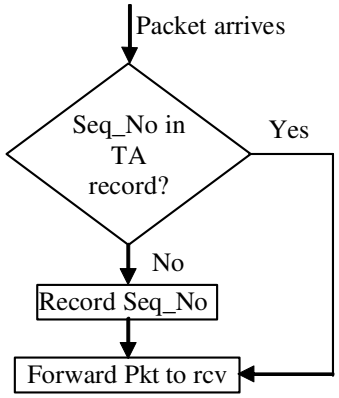

Fig. 3. Flowchart for data processing in BTS

The following is the ACK processing algorithm performed by the transit agent:

```
if (ACK_C-CLN arrives) then
   expt_seqno = get(SEQ_NO of expected packet);
   if (!isset(Fb)) then // if Fb hasn't been set
          if (pkt_transit[expt_seqno] == TRUE)  then
            set (Fb) = 1;
          else
            set (Fb) = 0;
          end if
   else
      If (pkt_transit[expt_seqno] == TRUE)  then
            set (Sb) = 1;
      else
            set (Sb) = 0;
      end if
   end if
end if
```

C. *Transport Protocol at Sender Side*

The E-TCP on the sender side has additional functionality to read the inserted information from the $ACK_{C\text{-}CLN}$ packet. Thus, it takes appropriate packet loss recovery action based on the information received. The current TCP (TCP Reno) is still required for the communication between devices on the fixed network[2]. Therefore, the protocol stack shown in Fig. 4 is implemented on the sender side.

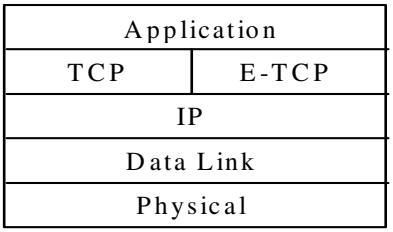

Fig. 4. Proposed Protocol Stack

Two transport protocols are available to maintain the reliability of data transmission between the two ends, depending on the type of network node at each

[2] In this case the sender does not need to use E-TCP since the wireless network is not involved in data transmission and the only reason for packet loss would be network congestion

end. Once the sender is able to distinguish the type of the receiver, the sender shall use TCP or E-TCP accordingly. The utilisation of both protocols is depicted in Table 1.

E-TCP will be active in the case when a wireless environment is involved in the data transmission process. An acknowledgment of the SYN packet with an Fb or Sb bit is an indication to the sender to activate the E-TCP or conventional TCP (Reno) as appropriate.

The following algorithm shows how E-TCP at the fixed sender learns about packet losses:

```
if (ACK_C-CLN arrives) then
  expt_seqno = get(SEQ_NO of expected pkt);
  if (Fb == 1) then // wireless loss detected
        retransmit_pkt();
        … keep ssthresh …
        … keep cwnd …
  else        // congestion loss detected
        … act like Reno …
        retransmit_pkt();
        … reset cwnd & ssthresh …
  end if
end if
```

The E-TCP on the fixed sender looks simple, since it only deals with a mobile receiver. But, it will be more complicated for a mobile sender, since the receiver could be a fixed host or another mobile host. The following is the E-TCP algorithm for a mobile sender:

```
if (ACK_C-CLN arrives) then
   expt_seqno = get(SEQ_NO of expected pkt);
   if (isset(Fb) && !isset(Sb)) then   // mobile to fixed host
         if (Fb == 1) then   // congestion loss detected
            … act like TCP Reno …
            retransmit_pkt();
            … reset cwnd & sshtresh …
         else     // wireless loss detected
            retransmit_pkt( );
         … keep ssthresh …
            … keep cwnd …
         end if
  else if (isset(Fb) && isset(Sb) then
     if {(Fb== 0 && Sb==0) | | (Fb == 1 && Sb == 1)} then
                  retransmit_pkt( );
                  … keep ssthresh …
                  … keep cwnd …
       else if  (Sb == 1 && Fb == 0) then
                  … act like TCP Reno …
                  retransmit_pkt();
                  … reset cwnd & sshtresh …
        end if
        end if
  end if
  end if
end if
```

D. Scenarios

In our proposed protocol stack, there are four possible communication scenarios that may occur between hosts as shown in Table 1. We shall discuss all four communication scenarios below:

1. *Fixed host to fixed host*

Data transmission between hosts in a fixed network has been discussed frequently and many improvements have been suggested; therefore we leave the standard version of TCP to handle the reliability of data transmission between fixed hosts. It is unnecessary to use E-TCP in this case, since a wireless network does not involve in data transmission between fixed hosts. Therefore, any packet losses could be assumed to be as a result of congestion loss.

2. *Fixed host to mobile host*

In this scenario, a fixed sender faces the challenge of distinguishing the cause of packet loss since the data travels over both fixed and wireless networks. TCP must be able to identify the location of the packet loss in order to maintain its performance. We discover that a base station that is located right in between the two different networks is a perfect location to assist the TCP sender distinguishing the cause of packet loss. Hence, we install an agent at the base station to perform a "transit checking" task.

Each time a packet transits the base station, the transit agent records the sequence number of the packet. This record shows that the packet has safely travelled over the fixed network, transited the base station and then the packet is forwarded to the receiver.

Upon receiving an acknowledgment ($ACK_{C\text{-}CLN}$) packet that contains a request for the expected packet, the agent checks its records against the sequence number of that expected packet. If the agent finds the sequence number in the record, it sets the "Fb" bit to 1. Otherwise, the agent sets the "Fb" bit to 0.

Upon receiving an $ACK_{C\text{-}CLN}$ packet, the sender learns which packet the receiver expects. Thus, it reads the Fb bit, if the value equals "1", the sender understands that the expected packet had transited the base station but it was not received by the receiver. This means that the expected packet was lost in the wireless network (due to corruption or, for some reason, other than congestion). On the other hand, if the value of Fb was "0", this means the expected packet has never transited this base station before. Therefore, it responds by indicating that the packet was actually lost in the fixed network (due to network congestion).

3. *Mobile host to Fixed host*

This scenario is quite similar to the data transmission scenario between the fixed host and the mobile host. The only difference is the interpretation of the Fb value. When the value of Fb is equal to "1", it means that the expected packet has previously transited the base station, thus the expected packet was lost in the fixed network. On the contrary, if the Fb

value is equal to "0" then the expected packet was lost in the wireless environment.

Fig. 5 shows the flowchart of $ACK_{C\text{-}CLN}$ processing in the base station.

4. Mobile host to Mobile host

Our proposed scheme also considers data transmission between mobile hosts. Since a packet must travel twice over the wireless network, the packet might transit through two base stations and travel once over a fixed network. In this case, there is the possibility of packet loss either in the wireless network or in the fixed network (Fig. 1).

In the case where packet loss occurs in the wireless area between the second base station (base station that serves the receiver) and the second mobile host (assume that this mobile host is the receiver), the sender may again assume that packet loss is due to congestion.

Therefore, the $ACK_{C\text{-}CLN}$ must allocate another bit namely the "Sb" bit which contains the transit history of the packet set by the second transit agent. In this case, the first base station reached will set its confirmation concerning the transit status in the Fb bit. When this $ACK_{C\text{-}CLN}$ reaches the second base station, the base station will assign a value for the Sb bit. The sender will be able to determine where the losses have occurred with information set in both the Sb and Fb. Table 2 identifies the different cases of Fb and Sb associated with the type of packet loss.

TABLE 2
FB AND SB VALUES ASSOCIATED WITH THE TYPE OF PACKET LOSS CASES

Bit	Sb = 0	Sb = 1
Fb = 0	Corruption	Congestion
Fb = 1		Corruption

A packet loss that is due to corruption should be retransmitted, but it is unnecessary for the sender to reduce its transmission rate (reduce the congestion window size). More discussions concerning each case can be obtained in [3].

Fig. 6 shows the flowchart for $ACK_{C\text{-}CLN}$ processing in the BTS for mobile-to-mobile communication.

VI. HANDOFF-RELATED PACKET LOSS

The Enhanced-TCP (E-TCP) and Transit Agent (TA) approach was discussed in [3] and [4], has shown improvement to TCP by distinguishing the two different types of packet losses (i.e. congestion or corruption) and to invoke appropriate recovery actions accordingly. We shall now consider another type of packet loss in the wireless environment known as handoff loss.

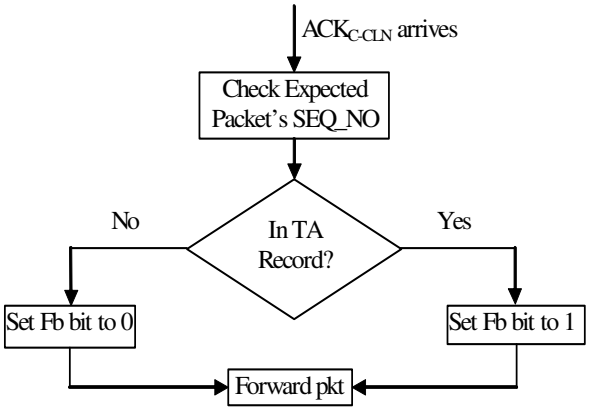

Fig. 5. Flowchart of ACKC-CLN Processing in BTS

In Fig. 7, let BTS1 represents the old base station and BTS2 represents the new base station. In the case when a data packet has been sent to a receiver through BTS1 while the mobile receiver has already been in the BTS2 coverage area, the mobile receiver may not receive the packet. Thus, the receiver will send an acknowledgment packet requesting the expected data packet. Since the TA on BTS2 does not have a transit record for the expected packet, TA will set the Fb bit to 0. This value suggests the existence of congestion loss to the sender. In fact, this was not the case.

A handoff-related packet loss should not be overcome using the TCP congestion avoidance mechanism [3][4][7][9][14]. This is due to the fact that there was no packet loss due to congestion in the network. Therefore, handoff loss should be recovered by retransmitting the lost packet without reducing the TCP transmission rate. Instead, handoff loss should be treated in a similar way to that which E-TCP handles corruption loss. Thus, there are no changes required for the transit agent and E-TCP.

A. Transit agent Synchroniser on base station controller

Due to the limitations of transit agents that have been proposed in previous works [3][4], we have been motivated to introduce the concept of a transit agent synchronizer (TASyn), which must be positioned at the base station controller (BSC)[3].

The TASyn performs a similar function to the transit agent. First, it records the sequence number of every transiting packet from the sender addressed to the mobile receiver on the currently active base station[4]. Second, it checks the transit history of the expected packet from its records for the old base station (assuming that the mobile host has already been in the coverage area of the new base station).

[3] A node that controls all the base stations, this node will have knowledge of the handoff process for mobile hosts.

[4] The base station that currently handles the data transmission

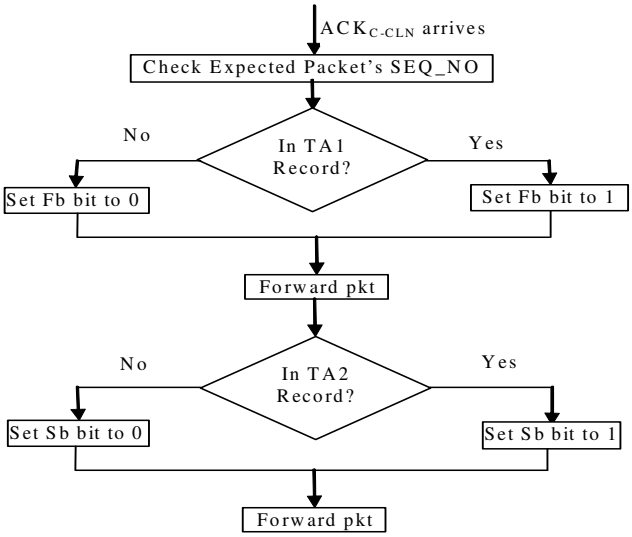

Fig. 6. Flowchart of ACKC-CLN for processing mobile-to-mobile communication

If the record was found, it explains that the expected packet was lost due to handoff (since the request for the expected packet came from the receiver via the new base station).

If the TASyn has a transit record of the sequence number of the expected packet while the value of Fb was set to 0 by TA at the new base station, TASyn will update the Fb bit to 1. The value is updated since TASyn understands that the handoff process has occurred.

The TASyn's task to update the Fb bit is over after all the handoff loss packets have been successfully retransmitted, but TASyn must continue to record the sequence numbers of any incoming data packet.

Multiple packets loss may occur during handoff process. Unfortunately, with the current implementation of TCP (Reno), it will trigger multiple reductions of the TCP congestion window size. Thus, it will significantly decrease TCP performance.

VII. SIMULATION

This section describes a simulation study for two situations: firstly, a simulation study for E-TCP to compare it with conventional TCP in order to distinguish congestion from corruption loss; in the second case, a simulation study for E-TCP including the implementation of TASyn to overcome handoff loss is presented.

The first case was simulated in three different scenarios as depicted in Fig. 1. The sender or receiver might be a fixed host or a mobile host. The base station includes a finite-buffer drop tail gateway and the network has both wired and wireless links. It is assumed that the sender always has data to send. It is also assumed that the receiver can always send out acknowledgments immediately for each data packet received without delay, other than the processing delay.

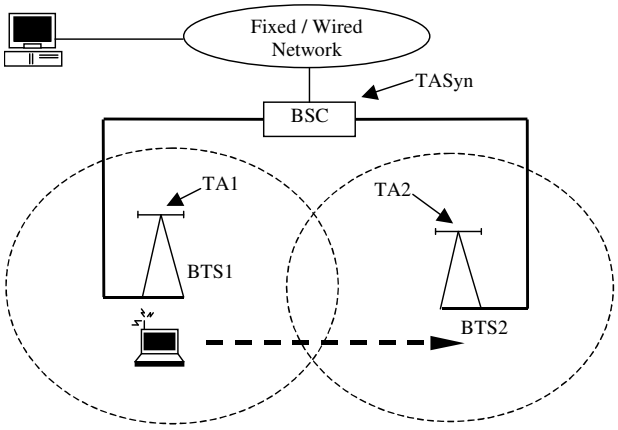

Fig. 7. Architecture for the Transit Agent Synchroniser

For simplicity, the simulations shown in this paper use a receiver that sends an ACK for every data packet received. The simulations also consist of one-way traffic. Therefore, ACK packets are never "compressed" or discarded on the path from the receiver back to the sender. The simulation models were developed with the well-known OPNET™ simulator.

The current implementation of TCP (TCP Reno) has been considered in this simulation. The simulation was performed using a typical wireless packet loss environment and suitable bit-error rate. We assume that the bit error rate for wired links is very low.

A. *Congestion and Corruption Loss*

TABLE 3 summarises some parameters used in our simulation model.

TABLE 3
PARAMETER SETTINGS

Wired link capacity	56 Kbps
Wired link propagation delay	30 msec
Wired link BER	10^{-9}
Wireless link capacity	2 Mbps
Maximum window size	94
TCP segment size	170 bits
Minimum slow-start threshold	1
Bit error rate	10^{-3} to 10^{-6}

In order to measure the performance of TCP over a wireless link, we simulated in all models using a range of different values for the bit-error rate (TABLE 3). All results from our simulations are obtained using standard 95% confidence levels. We have performed our simulation using three different scenarios. The three scenarios cover the following cases: We first assume that data was sent from the fixed host to the mobile host. In the second case data is transmitted from the mobile host to the fixed host and finally, data transmission from mobile to a mobile host. In all of

these cases we assume that the mobile host stays in one cell during the lifetime of the connection. We then report the simulation results under varying bit-error conditions over the wireless channel, where the ranges of BERs are from 10^{-3} to 10^{-6}.

TABLE 4
NUMBER OF RECEIVED PACKET AT RECEIVER

direction / protocol	F to M	M to F	M to M
E-TCP	3176	3341	3459
Reno-TCP	2624	2722	2735
%	21.04	22.74	26.47

Performance of E-TCP and Reno on data transmission from fixed host to mobile host is depicted in Fig. 8. It shows that E-TCP performance is slightly better than Reno as the BER is increased. E-TCP allows more packets to be received. In addition, the sender was also able to maintain its transmission rate due to its new ability to distinguish between the different types of packet loss. Table 4 shows that E-TCP sends, on average, about 21.04% more packets compared to Reno-TCP. The reason is that E-TCP can maintain its congestion window size. On the contrary, TCP Reno still assumes that the packet losses were due to congestion. Thus, it reduces its congestion window size (transmission rate) after the lost packet was retransmitted and the reduction in the window size results in fewer new packets being sent.

Fig. 9 shows goodput comparisons between E-TCP and TCP Reno on data transmission from mobile host to fixed host. It also shows that E-TCP can provide a better goodput than Reno. With the new functionality incorporated into E-TCP, it is able to avoid unnecessary invocation of the congestion control mechanism. On average, we see that about 22.74% more packets have been successfully received with the implementation of E-TCP.

Finally, Fig. 10 shows the goodput comparison for different values of BER whilst using E-TCP and Reno-TCP for data transmission between mobile hosts. Our simulation results show that 26.47% more packets were received successfully at the receiver.

Another set of simulations was run with similar settings; except that the queue size was limited to 50 packets only. The wired link capacity was set to 100 Mbps and the wireless BER used was 10^{-3}. Fig. 11 shows the results of a throughput comparison when Reno or E-TCP was implemented in the three different scenarios. It shows that E-TCP performance is better than Reno.

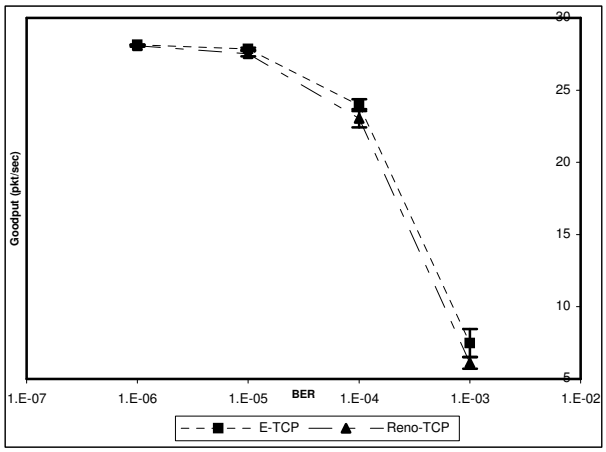

Fig. 8. Goodput comparisons at different BER on data transmission from fixed to mobile host

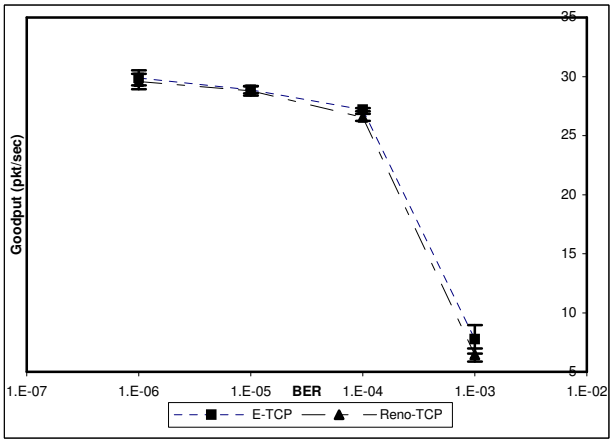

Fig. 9. Goodput comparisons at different BER on data transmission from mobile to fixed host

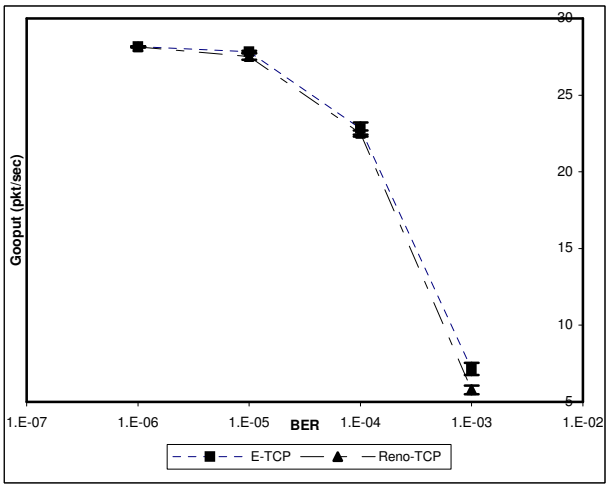

Fig. 10. Goodput comparisons at different BER on data transmissions from mobile to mobile host

Fig. 12 (a) and (b) shows the trace of congestion window size (cwnd) for E-TCP and Reno for data transmission from a fixed to a mobile host. From the figure, we can see that E-TCP can maintain a larger congestion window size compared with Reno. Since Reno is unable to distinguish the type of losses, it