Hierarchical Cache Design for Enhancing TCP over Heterogeneous Networks with Wired and Wireless Links

Jian-Hao Hu, Kwan L. Yeung
Department of Electrical & Electronic Engineering
The University of Hong Kong
Hong Kong, China
E-mail: kyeung@eee.hku.hk

Siew Chee Kheong and Gang Feng
ICIS, School of EEE
Nanyang Technological University
Singapore
{ecksiew,gfeng}@ntu.edu.sg

Abstract- **In this paper, we propose a two-layer hierarchical cache architecture for enhancing TCP performance over heterogeneous networks with both wired and wireless links. A new network-layer protocol, called New Snoop, is designed. The main idea is to cache the unacknowledged packets at both Mobile Switch Center (MSC) and Base Station (BS), thus forming a two-layer cache hierarchy. If a packet is lost due to transmission errors in wireless link, the BS takes the responsibility to recover the loss. When a handoff occurs during a TCP connection session, the packets cached in MSC can help to minimize the latency of retransmissions due to temporal disconnection. Simulation results show that using New Snoop is significantly more robust in dealing with unreliable wireless links and handoffs as compared with the Snoop scheme [3] as well as other existing TCP enhancements.**

I. INTRODUCTION

The two growing technology trends, Internet access and wireless access, would result in a strong combination that wireless data networks and services would dominate the marketplace. TCP is a transport layer protocol that has been designed, improved and tuned to work efficiently on wired network where the packet loss is very small. Whenever a packet is lost, it is reasonable to assume that congestion has occurred on the connection path. Hence, TCP triggers congestion recovery algorithms when packet loss is detected. On the other hand, the bit error rate of a wireless link is much higher and a wireless connection might be temporally broken due to a handoff or other temporal link impairment such as shadowing effect. As a result, the assumption that packet loss is (mainly) due to congestion is no longer valid and the original TCP cannot work well in a heterogeneous network with both wired and wireless links. In summary, the challenges for TCP over wireless networks are: (1) high bit error rates, (2) handoff due to users' mobility, (3) asymmetric effects and latency variation, and (4) low channel bandwidths.

Since many network applications are built on top of TCP, and will continue to be in the foreseeable future, it is important to improve its performance in wireless networks without any modifications to the fixed hosts. This is the most promising way by which mobile devices can seamlessly integrate with the rest of the Internet. Recently, several reliable transport-layer protocols for networks with wireless links have been proposed to alleviate the poor end-to-end TCP performance in the heterogeneous network environments, including Split Connection Approach [2], Fast-Retransmit Approach [4], Link-level Retransmissions [1] and Selective Acknowledgements [5]. However, these schemes have some limitations with respect to one or more of the following aspects: TCP semantics transparency, application relinking requirement, software overhead or others.

It is worthy noting that a good scheme called Snoop was proposed in [3]. The Snoop protocol introduces a module, called snoop agent, at the base station. The agent monitors every packet that passes through the TCP connection in both directions and maintains a cache of TCP packets sent across the link that have not yet been acknowledged by the receiver. The snoop agent retransmits the lost packet if it has it cached and suppresses the duplicate acknowledgments (ACKs). Like other link-layer solutions, the snoop approach could also suffer from not being able to completely shield the sender from the wireless losses.

In this paper, we propose a two-layer hierarchical cache architecture for enhancing TCP performance over heterogeneous networks with both wired and wireless links. A new network-layer protocol, called New Snoop, is designed. The main idea is to cache the unacknowledged packets at both Mobile Switch Center (MSC) and Base Station (BS), thus forming a two-layer cache hierarchy. If a packet is lost due to transmission errors in wireless link, the BS takes the responsibility to recover the loss. If the loss/interruption is due to a handoff, the MSC performs the necessary recovery. With this proposed hierarchical cache architecture, New Snoop protocol can effectively handle the packet losses caused by both handoffs and link impairments. This in turn improves the TCP throughput performance as compared with the Snoop scheme [3] as well as other existing TCP enhancements.

In this paper, we consider the case where a fixed host, e.g. an ISP server, using TCP to communicate with a mobile host via a path consisting of an error-immune wired link and an error-open wireless link. In the next section, the two-layer hierarchical cache architecture and the New Snoop protocol are proposed. The simulation model and simulation results for evaluating the New Snoop protocol are presented in Section III. Finally we conclude the paper in Section IV.

II.TWO-LAYER HIERICHICAL CACHE ARCHITECTURE AND NEW SNOOP

Using the proposed two-layer hierarchical cache scheme, no modifications to the two end hosts and the fixed network are required except at the MSC and BS. At these two points, the routers are modified to cache the unacknowledged packets going to mobile hosts using a new network-layer protocol called New Snoop. Local retransmission over the wireless link is performed by the BS when a packet loss due to wireless transmission errors is detected. If a connection interruption due to a handoff occurs, the MSC performs the necessary recovery. We can show that the sender can be completely shielded from the instabilities of the wireless links and the user mobility.

A. New Snoop Protocol Layer

Like Snoop protocol [3], New Snoop is located above the IP layer. Its function is to monitor and cache the packets passing through the BS/MSC, to abstract the code information from the TCP packets, and to utilize this information for local packet recovery. New Snoop differs from the original Snoop in four major aspects. With New Snoop, (1) packets are cached at both MSC and BS, (2) packet loss due to congestion at the BS is considered, (3) local fast retransmission of out-of-order packets is implemented, and (4) each BS has an ACK adjustment mechanism to handle the latency variation caused by local retransmission.

According to our understanding of Snoop in [3], the buffer for each TCP connection at the BS is divided into two portions, we call them *output buffer* and *cache buffer* as shown in Fig. 1. A packet arrives and waits in the output buffer for transmission to the mobile host. Upon transmission, a copy of the transmitted packet is stored at the cache buffer until the ACK that confirms its correct reception by the mobile host is received. Since the size of the cache buffer is fixed, a transmitted packet cannot be cached if the cache buffer is full. Therefore using Snoop *does not* guarantee a packet lost on the wireless link can be found at the BS. If this happens, the BS will forward the received duplicate ACKs to the original TCP sender and subsequently a fast retransmission at the sender for recovering the loss will be triggered. *In this case, the loss caused by wireless transmission errors will be wrongly interpreted by the sender as traffic congestion.*

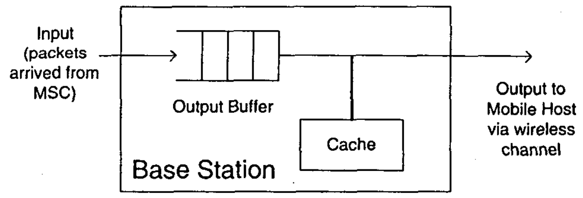
Input (packets arrived from MSC)
Output Buffer
Cache
Base Station
Output to Mobile Host via wireless channel

Fig. 1 Buffer design at the BS for a TCP connection using Snoop protocol.

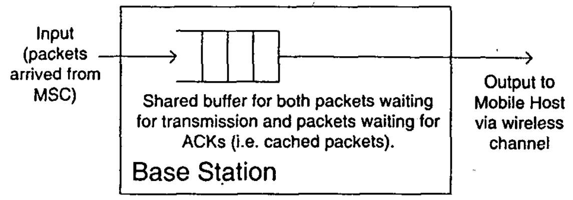
Input (packets arrived from MSC)
Shared buffer for both packets waiting for transmission and packets waiting for ACKs (i.e. cached packets).
Base Station
Output to Mobile Host via wireless channel

Fig. 2 Buffer design at the BS for a TCP connection using New Snoop protocol.

Using New Snoop, we propose to use a shared buffer for both packets waiting for transmission and packets waiting for ACKs as shown in Fig. 2. When a new packet arrives at the BS, it is discarded if this shared buffer is full. Similar to a conventional router, a buffer overflow implies that traffic congestion occurs. (The congestion effect on Snoop is not considered in [3].) Upon transmission, a copy of the transmitted packet *remains* in the buffer but becoming a cached packet (a packet tag for distinguishing the packet status will be described later). The cached packet is cleared from the buffer when the ACK confirmed its correct reception is received. Therefore if a packet is lost due to errors in wireless link, it can *always* be found in the BS buffer. The local loss recovery can then be performed and the original sender will be completely shielded from the instabilities of the wireless links. Another advantage of using shared buffer is that the buffer utilization will be greatly improved because complete resource sharing policy is used.

B. Packet Loss Recovery due to Wireless Transmission Errors

Recall that the TCP protocol is a sliding window scheme implemented at the sender. Each TCP packet has an associated sequence number. A TCP packet is identified uniquely by the sequence number and its size. For implementation, we propose to add a three-bit tag to each packet arrived/buffered at the BS. This tag is used only at the BS to indicate the status of the buffered packet, where the first bit of the tag denotes the associated packet is waiting for transmission <0> or waiting for ACK <1>; the second bit denotes the packet is an out-of-order packet <1> or an in-order packet <0>; and the third bit denotes the packet is a retransmitted packet <1> or not <0>.

When a new packet arrives at the BS, it is assigned a tag with value 000 or 010, denoting the packet is waiting for its turn of transmission (to the mobile host). Tag 000 is used if it is an in-order packet, meaning that its sequence number is one plus that of the last received packet. Tag 010 is used if the new packet is an out-of-order packet. Upon transmission, the tag of the transmitted packet is converted to 100 or 110 to denote that this packet is a cached copy. It remains in the buffer until an ACK confirmed its correct reception by the mobile host is received. Finally, a packet is marked as a retransmitted packet with tag xx1 (where x means *don't care*, it can be either 0 or 1) if it has been retransmitted by the sender or by the BS. Depending on the tag value, the BS carries out the corresponding packet processing procedure and/or ACK processing procedure as described below.

B1. Packet Processing at BS

1) When an in-sequence packet arrives and found no free buffer at the BS, the packet is immediately discarded. Otherwise, attach tag 000 to the packet and place the packet to the transmission queue. When the packet is transmitted via the wireless link later, a timestamp is attached to its cached copy (for measuring the round-trip-time of the wireless loop as described later) and its tag becomes 100.

2) When an out-of-order packet that has not been cached before (i.e. no copy of this packet can be found at the BS) arrives and local buffer is available, attach tag 010 to the packet and place the packet to the transmission queue. Upon transmission, its tag becomes 110. It should be noted that although this packet is out-of-order, there is no need to give this packet a higher transmission priority over other packets waiting in the transmission queue.

3) When an out-of-order packet that has not been cached before arrives and local buffer is not available, if the sequence number of the new packet does not equal to the sequence number of the last received ACK, the packet is discarded. Otherwise, *the mobile host is expecting this packet.* That means the mobile host is waiting for this packet in order to generate an ACK to release

some packets from the fully occupied BS buffer, and the wireless link is idle at this time. The new packet is immediately transmitted to the mobile via the idle wireless channel, and a copy of it is stored at a special one-packet buffer (which is not explicitly shown in Fig. 2). In this case, it is reasonable to assume that this packet has been lost previously in the fixed network, and right now it is a retransmitted copy of the previously lost packet. Therefore a tag 111 is attached to this cached packet. We refer this as *local fast retransmission of out-of-order packet.*

4) When an out-of-order packet arrives and a copy of this packet is found in the BS buffer, this newly arrived duplicate packet is immediately dropped. This happens when some previous packet corrupted and caused a fast packet retransmission. The long delay for recovering the previously lost packet causes the current packet (i.e. the one just arrived) to timeout at the sender (since no ACK for it was issued by the mobile host). However, the previous copy of this packet has already been transmitted to the mobile host and the BS is waiting for the corresponding ACK.

B2. ACK Processing at BS

New Snoop processes ACK with two major differences from Snoop [3]. First, using Snoop all out-of-order packets are marked as retransmitted packets. If an out-of-order-packet is lost in the wireless link, the BS must wait for local retransmit timeout before triggering a local packet retransmission. If the wireless link transmission rate is high (e.g. 2Mbps or 384 kbps) and the lower layer protocol processing time is long, the throughput degrades quickly while waiting for timeout.

Using New Snoop an out-of-order packet is judged as lost at the wireless link when the 3rd duplicate ACK for it arrives at the BS. (Note that for an in-order packet, the *first* duplicate ACK signifies that this packet has been lost in the wireless link and a local retransmission can thus be immediately triggered.) The BS retransmits the packet immediately and the successive duplicate ACKs are held by the BS (i.e. without forwarding to the sender) until a new ACK is received. *Using this approach, the sender will only receive two duplicate ACKs for the packet lost at the wireless link and that would not inappropriately (like in Snoop) trigger fast retransmission at the sender.*

The local loss recovery at the BS will cause extra delay (and delay variation) for ACKs to reach the sender. The length of this extra delay is a function of the packet loss probability on the wireless link. It is important to have a *good* estimation of the round trip delay time at the sender. This can avoid the inaccurate setting of the retransmit timeout (*RTO*) value and thus unnecessary retransmission by the sender. To achieve this, the BS should have a mechanism to adjust the forwarding time of each ACK towards the sender. The detail of the ACK adjustment is described in the next sub-section. This is the second important difference with Snoop.

The following pseudo code describes the steps when an ACK is received by the BS. (The ACK adjustment scheme is not included.)

```
(1) When a new ACK is received, the acknowledged packets in the
    buffer (including the one-packet special buffer) are flushed. The
    round trip time estimate for the wireless link is updated. The
    same ACK is then forwarded to the sender.
(2) When a spurious ACK is received, it is discarded without
    forwarding. A spurious ACK is an ACK with a sequence number
    less than that of the last received ACK.
(3) When a duplicate ACK is received,
       if the requested packet is not found in the buffer, forward the
       duplicate ACK to the sender.
         if the requested packet is found in the buffer,
           if its tag is 100 (i.e. it is an in-order packet), retransmit
           the requested packet and reset the retransmit timer. Then
           mark the packet as retransmitted with tag xx1, where x
           means either 0 or 1. Hold the duplicate ACK.
           if its tag is 110 (i.e. it is an out-of-order packet),
             if DUP_counter < 3, DUP_counter = DUP_counter + 1.
             if DUP_counter = 3, retransmit the requested packet and
               reset the retransmit timer. Then mark the packet as
               retransmitted with tag xx1. Hold the duplicate ACK.
               Reset DUP_counter = 0.
           if its tag is xx1 (i.e. it is a retransmitted packet), hold the
           duplicate ACK.
 (4) When the retransmit timer times out, check the sequence
       number of the associated timeout packet. If its sequence
       number is the same as that of the last received ACK,
       retransmit the timed out packet and reset the retransmit timer.
       Mark the packet as retransmitted with tag xx1.
```

DUP_counter is an integer counter that counts how many duplicate ACKs have been received if the requested packet is an out-of-order packet.

C. ACK Adjustment for Handling Latency Variation

The TCP sender estimates the RTT (round-trip-time) of a connection by measuring the time between the sending of a packet and the receipt of the ACK for this packet. For each congestion window *CWND* the TCP sender can get a sample of the RTT. The new *RTT* and its deviation are calculated below:

$$RTT = \alpha \times RTT + (1-\alpha) \times M \qquad (1)$$

$$D = \alpha \times D + (1-\alpha) \times |RTT - M| \qquad (2)$$

where D is the mean deviation of *RTT*; M is the sample of the *RTT*; α is the control factor. The optimal value of α (as suggested by RFC of IETF) equals to 0.875 in order to have small oscillation between two *RTT*s.

In TCP protocol, the retransmit timeout (*RTO*) value of the sender is calculated as a function of the average and mean deviation of the *RTT*, where

$$RTO = RTT + 4 \times D \cdot \qquad (3)$$

Most TCP implementations use a 500 ms timer granularity for the retransmission timeout. In heterogeneous networks, the local retransmission at the wireless link can cause the end-to-end *RTT* to increase significantly in a short time due to the long processing time and the low data transmission rate on the wireless link. As a result, big time gaps in the receiving ACK sequence are observed at the sender when local retransmission occurs. Frequent wrong timeouts at the sender would then be triggered. Remember that wrong timeout has a serious negative impact on the TCP connection throughput.

In New Snoop, we propose to use an ACK adjustment scheme to solve this problem. Each ACK that is to be forwarded to the sender will be held or delayed for a short period of time at the BS. As a result, the sender will get a longer but more stable (i.e.

with less delay variation) *RTT* measurement, and the probability of wrong timeout at the sender can be reduced. The ACK delay interval, Δ_{Delay}, is calculated as a function of the round trip time and the packet loss probability of the wireless link, or

$$\Delta_{Delay} = P_{Pkt\_loss} \times RTT_{wireless} \qquad (4)$$

where P_{Pkt_loss} is the packet loss probability of the wireless link; $RTT_{wireless}$ is the round trip time of the wireless link. $RTT_{wireless}$ is measured by the timestamp of each packet in the local buffer at the BS using the same formula in equation (1) above.

D. Fast Packet Recovery during Handoffs

When a handoff occurs during a TCP connection session, the transmission channel is temporally not available and TCP packets are lost. A handoff latency may last for several tens ms or higher. Packet losses due to handoffs can cause the TCP sender to remain idle for long period of time even after the handoff is completed, resulting in unacceptably low throughput.

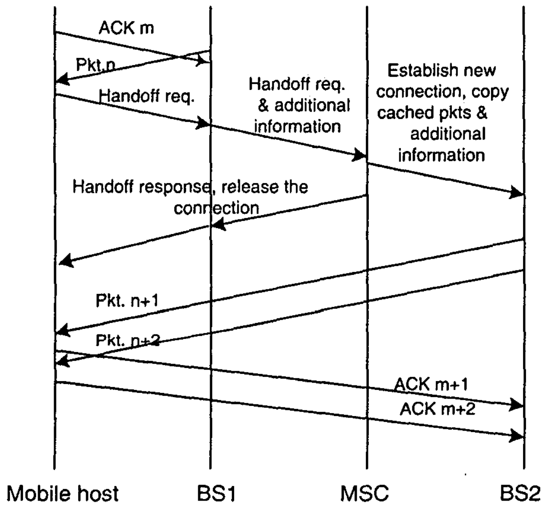
ACK m
Pkt.n
Handoff req.
Handoff req. & additional information
Establish new connection, copy cached pkts & additional information
Handoff response, release the connection
Pkt. n+1
Pkt. n+2
ACK m+1
ACK m+2
Mobile host
BS1
MSC
BS2

Fig. 3 Handoff processing in two-layer hierarchical cache architecture using New Snoop.

In New Snoop, we propose to tackle this by storing unacknowledged packets at MSC for fast local loss recovery. For simplicity, we can assume that the buffer sizes at both BS and MSC are equal for each TCP connection. Fig. 3 shows a typical scenario of packet transmissions during a handoff in a network using two-layer hierarchical cache scheme. When a mobile host finds a base station with a stronger signal power, a handoff request is sent to MSC via the old base station (BS1). The MSC then re-routes the TCP connection to the new base station (BS2). The MSC keeps track which packets have been sent to BS1 during the handoff. It collects the following information from BS1: (a) which packets are waiting for ACKs from the mobile host, and (b) the associated local retransmit timer values for packets waiting for ACKs. Then the MSC sends all unacknowledged packets from its cache buffer plus their retransmit timers collected from BS1 to BS2. Then BS2 takes over the role of BS1. The handoff completes and the TCP connection is resumed.

The buffer requirement at a MSC can be greatly reduced if a handoff can be predicted. In fact, predicting a handoff is not difficult. We can, for example, start to cache the packets arrived at the MSC only when a handoff request from a mobile is received. If this duration is not enough for caching at least the same amount of unacknowledged packets as the old base station (BS1), we can adjust the handoff initiation power threshold such that a handoff request can be initiated earlier, or the MSC will be warned about the handoff earlier.

III. PERFORMANCE EVALUATION

In this section, we study the performance of the New Snoop under the hierarchical cache architecture using computer simulations. Fig. 4 shows our simulated network, which consists of an ISP server, a MSC, two BSs and a mobile host. We assume that the mobile host is moving in one direction from left to right. A handoff occurs at the boundary between two cells that are covered by two adjacent base stations. The simulation parameters are summarized in Table 1.

We assume that the links in the fixed network are error-free, and the congestion in the fixed network only occurs at the BS. We also assume that some error-correction schemes have been implemented at physical and data link layers, therefore the packet lost probability as seen by the TCP layer is a uniformly distributed random process. Since ACK packets are relatively small in size, the effect of ACK loss is ignored.

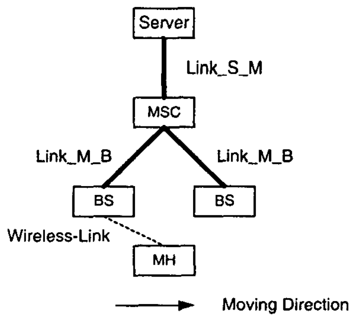

Fig. 4 Simulation Model.

For comparison, TCP Reno, New Reno, Reno with Snoop/New Snoop, and New Reno with Snoop/New Snoop are implemented. For all protocols except those with Snoop option, the base station is assumed to have a fixed (shared) buffer size of $Size_{cache}$= 20 packets. For protocols with Snoop option, we adopt the buffer design shown in Fig. 1. That is the output buffer is $Size_{cache}$= 20 packets, and the extra cache buffer is assumed to be infinite.

A. Performance in Recovering Losses caused by Wireless Transmission Errors

We first study the performance of New Snoop (NS) protocol in recovering losses caused by wireless transmission errors. The packet loss rate (PLR) of the wireless link under consideration is from 5×10^{-1} to 5×10^{-5}. We simulate the case that a mobile host retrieves a buck data file from the fixed server. The size of the download file is 50 Mbytes for minimizing the effect of TCP slow start on the throughput performance.

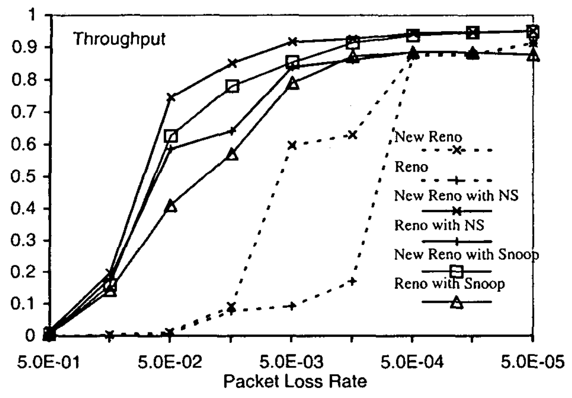

Fig. 5 Recovering losses caused by wireless link errors; 384 Kbps wireless link.

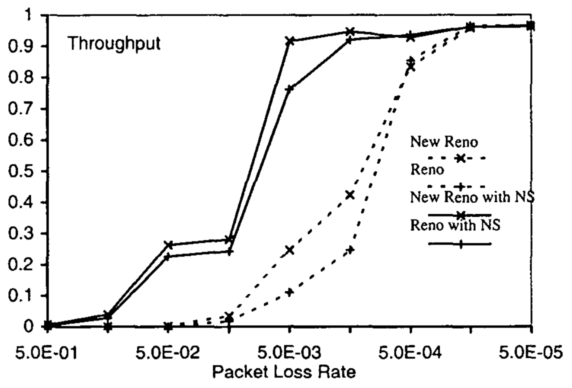

Fig. 6 Recovering losses caused by wireless link errors; 2 Mbps wireless link.

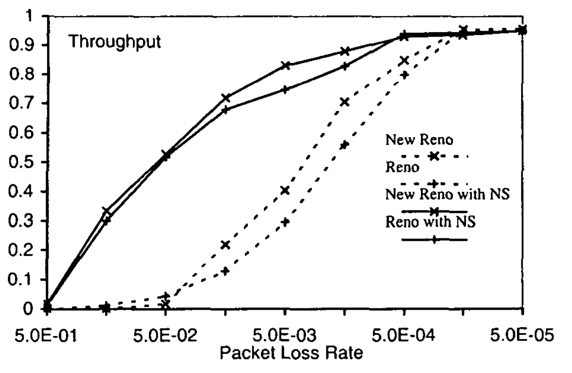

Fig. 7 Recovering losses caused by wireless link errors; 144 Kbps wireless link.

Figs. 5, 6 & 7 show the throughput of wireless links with transmission rates 384 kpbs, 2Mbps and 144 kbps respectively. We can see that using New Snoop can significantly improve the throughput when PLR is higher than 5×10^{-4}. This indicates that the packet losses due to errors at the wireless link have been effectively recovered by New Snoop. From PLR of 10^{-1} to 10^{-2}, the throughput improvement obtained using New Snoop is from 10 ~ 60 times as compared to that of the (pure) TCP Reno and TCP New Reno in Fig. 5, 8 ~ 67 times in Fig. 6, and 3 ~ 23 times in Fig. 7.

From Fig. 5, we can see that the throughput of using New Snoop is higher than Snoop. (Remember that we have used an infinite sized cache buffer for Snoop. Therefore if finite sized cache buffer is used, Snoop will perform even worse.) For PLR from 10^{-1} to 5×10^{-3}, about 4~12% and 3~17% throughput improvements are obtained when New Reno and Reno are used respectively. (We also notice that the improvement using Reno is more than New Reno. This is because if multiple packets are lost due to buffer overflow, Reno has a higher probability of causing sender timeout. When packet loss due to wireless PLR is low, the performance difference between using Reno and New Reno are small.)

Comparing Figs. 6 & 7, it is interesting to see that the throughput using 2Mbps is lower than that of the 144kbps when PLR is high, while the vice versa is true when PLR is low. This is because when the wireless link rate is high, there are more unacknowledged packets in the local cache at the BS. If the PLR of the wireless link is high, the local cache overflows with a higher probability. Then the TCP sender will enter the fast retransmission and fast recovery procedures to recover the lost packets. This results in a significant drop in throughput. On the other hand, when the PLR is low enough, the local cache overflow is less likely. Therefore, the link with a higher transfer rate can achieve a higher throughput.

Table1 Simulation Parameters

Parameters	Symbols	Values
Transfer rate of the Link between the Server and the MSC	$R_{Link\_S\_M}$	10 Mbps
Transfer latency of the Link between the Server and the MSC	$L_{Link\_S\_M}$	40 ms
Transfer rate of the Link between the MSC and the BS	$R_{Link\_M\_B}$	10Mbps
Transfer latency of the Link between the MSC and the BS	$L_{link\_M\_B}$	1 ms
Transfer rate of the Wireless Link	$R_{Wireless}$	2Mbps / 384Kbps / 144 Kbps
Transfer latency of the Wireless Link	$L_{Wireless}$	20 us
Packet Loss Probability of the Wireless Link	P_{pkt_loss}	5×10^{-1} ~ 5×10^{-5}
Processing delay of the lower layer protocols at the wireless link	$D_{process}$	100 ms
Handoff interval	Handoff	30s ~ 180s
Buffer size of the MSC and BS for each TCP connection	$Size_{cache}$	20 Packets
TCP packet size	$Size_{Pkt}$	512 Bytes
ACK size	$Size_{ACK}$	40 Bytes

B. Performance in Recovering Losses during Handoffs

Next we focus on the performance of using New Snoop in handling interruptions caused by handoffs. Assume the mobile host moves at a constant speed from left to right as shown in Fig.

4. The handoff frequency or the interval between two handoffs depends on the mobile moving speed. In our simulations, we assume handoff occurs at a regular interval from 30 seconds to 180 seconds. For comparison, the handoff performance of the Snoop protocol is also obtained. The handoff processing time is assumed to be 20 ms for New Snoop using the hierarchical cache architecture, and 20 ms plus the packet transmission time (from the old BS to the New BS) for the Snoop protocol. The packet loss due to handoff is not considered for Snoop protocol.

Two types of wireless environments are simulated (Figs. 8 & 9), one with a high PLR (=5×10^{-2}), and another with a low PLR (= 5×10^{-3}). For each type of environment, three wireless link transfer rates, 144kbps, 384kbps and 2Mbps, are used. In all simulations, the size of the download file is 50 Mbytes and TCP New Reno is adopted.

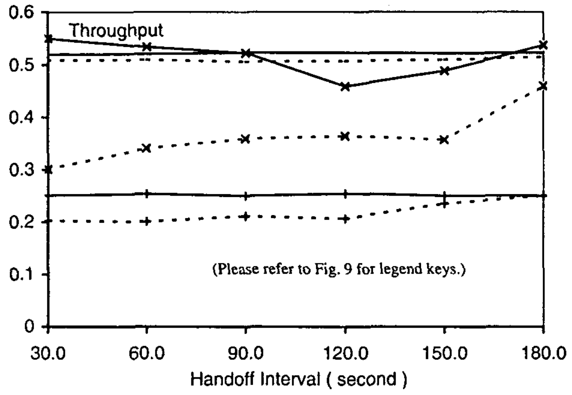

Fig. 8 Performance of New Snoop with hierarchical cache to recover from handoffs; PLR = 5×10^{-2}.

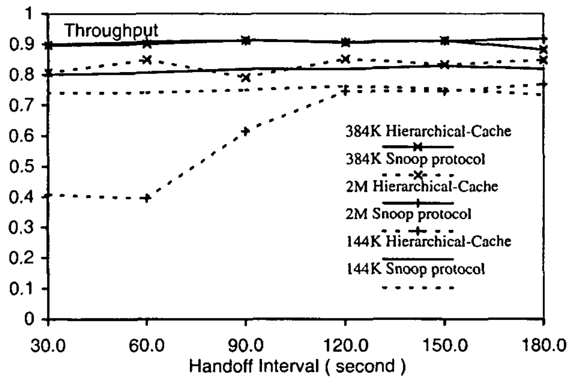

Fig. 9 Performance of New Snoop with hierarchical cache to recover from handoffs; PLR=5×10^{-3}.

From Fig. 8, we can see that New Snoop with the hierarchical cache scheme improves the wireless link throughput by about 20%, 5% and 2% as compared to that of using Snoop protocol at rates 384 kbps, 2 Mbps and 144kbps, respectively. Compared with Figs. 5, 6 & 7 with the same PLR, we find that handoff has a stronger performance impact to New Snoop for the case with 384 Kbps rate. The throughput drops about 20%, while the throughput drops for 2 Mbps and 144 kbps are only about 1% and 0.5%.

Fig. 9 shows the case with wireless link PLR = 5×10^{-3}. We can see that hierarchical cache scheme improves the throughput by 5%~10%, 15%~50% and 6%~7% as compared to that of using Snoop, at rates 384 kbps, 2 Mbps and 144kbps respectively.

IV. SUMMARY

In this paper, we have proposed a two-layer hierarchical cache architecture for enhancing the TCP performance in a heterogeneous network with both wired and wireless links. A new network layer protocol called New Snoop was designed for implementing at both base station (BS) and mobile switching center (MSC). The idea to cache packets at these two places for effectively recovering retransmissions caused by wireless link errors as well as handoffs. The simulation results showed that the proposed scheme is robust and can significantly improve wireless link throughput when packet loss rate is high, and the improvement for high-speed wireless links is more significant than that for low-speed links.

ACKNOWLEDGEMENTS

This work was supported by Competitive Earmarked Research Grant HKU 1180/00E and ZhongXing Telecom Research Fund.

References:

[1] E. Ayanoglu, S. Paul, et.al, "A Link-Layer Protocol for Wireless Networks," ACM/Baltzer Wireless Networks Journal, 1:47--60, February 1995.

[2] A. Bakre and B. R. Badrinath, "I-TCP: Indirect TCP for Mobile Hosts," Proc. 15th Intern. Conf. on Distributed Computing Systems (ICDCS), May 1995.

[3] H. Balakrishnan, S. Seshan, and R.H. Katz, "Improving Reliable Transport and Handoff Performance in Cellular Wireless Networks," ACM Wireless Networks, 1(4), December 1995.

[4] R. Caceres and L. Iftode, "Improving the Performance of Reliable Transport Protocols in Mobile Computing Environments," IEEE Journ. on Sel. Areas in Comm., 13(5), June 1995.

[5] K. Fall and S. Floyd, "Simulation-based Comparisons of Tahoe, Reno, and Sack TCP," Computer Communications Review, 1996.