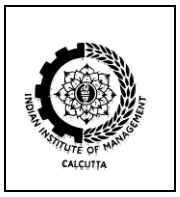

INDIAN INSTITUTE OF MANAGEMENT CALCUTTA

WORKING PAPER SERIES

WPS No. 649/ January 2010

A New Protocol to improve TCP Performance in Network Mobility

by

Bhaskar Sardar
Department of Information Technology, Jadavpur University, Kolkata, India

Debashis Saha
Professor, IIM Calcutta, Diamond Harbour Road, Joka P.O., Kolkata 700 104 India

&

Mahbub Hassan
School of Computer Science & Engg, University of New South Wales, Sydney, Australia

**** A part of this paper was presented in IEEE Vehicular Technology Conference 2006***

A New Protocol to improve TCP Performance in Network Mobility *

Bhaskar Sardar
Department of Information Technology
Jadavpur University
Kolkata, India
bhaskargit@yahoo.co.in

Debashis Saha
MIS and Computer Science Group
IIM Calcutta, Kolkata
Kolkata, India
ds@iimcal.ac.in

Mahbub Hassan
School of Computer Science and Engineering
University of New South Wales
Sydney, Australia
mahbub@cse.unsw.edu.au

Abstract: *Network mobility (NEMO) is a mobility management solution that allows various types of moving networks, e.g. network of sensors deployed in a vehicle, to be permanently connected to the Internet. An onboard mobile router (MR) connects the moving network to the wired infrastructure by means of high-speed cellular or any other wide area mobile data services. One application of NEMO attracting commercial interest is the deployment of wireless local area networks inside public transport vehicles, e.g., trains and buses, to provide Internet access to passengers. However, unlike the traditional terminal mobility, where the mobile hosts (MHs) connect to the cellular base station directly, passengers using the NEMO solution encounter an additional wireless link (MR-MH) before their MHs get connected to the wired infrastructure. In this paper, we analyze and quantify the impact of the additional wireless link on the performance of the widely used TCP protocol. Our analysis reveals that TCP performance schemes designed for conventional terminal mobility are not as effective in network mobility. We propose on-board TCP (obTCP) to effectively address the wireless link related issues in network mobility. We compare its performance against a classical scheme, called snoop, known for its effectiveness in terminal mobility. Using analytical means we demonstrate that the performance gain of obTCP over snoop increases linearly with delays and exponentially with the loss probabilities in the wireless links. These analytical observations are validated through extensive ns-2 simulations. We then extend these analyses to obtain throughput models of snoop and obTCP in NEMO. Our simulations further demonstrate that obTCP can coexist with snoop in the same infrastructure (e.g., cellular base stations) without causing serious unfairness to each other.*

Keywords*: NEMO, wireless TCP, snoop, mobile router, performance analysis.*

**** A part of this paper was presented in IEEE Vehicular Technology Conference 2006***

I. Introduction

Figure 1 shows the connectivity model of various mobile communication elements when Internet hot spots are offered on public transport vehicles, e.g., trains and buses [1]. Passenger devices connect to an onboard wireless local area network (WLAN), which remains connected to the fixed host (FH) via a mobile router (MR). The MR manages the Internet connectivity of the entire vehicle by making use of any available wireless carrier networks, cellular, satellite, or even a roadside WLAN, in the background. This form of mobile communication, where an entire network is treated as a single mobile unit, is often referred to as *network mobility* or NEMO. The Internet Engineering Task Force (IETF) has recently released standards [2] supporting a MR-based solution for connecting any type of moving networks, including a personal area network (PAN), to the fixed Internet. The release of the standards is expected to accelerate the commercial deployment of this technology.

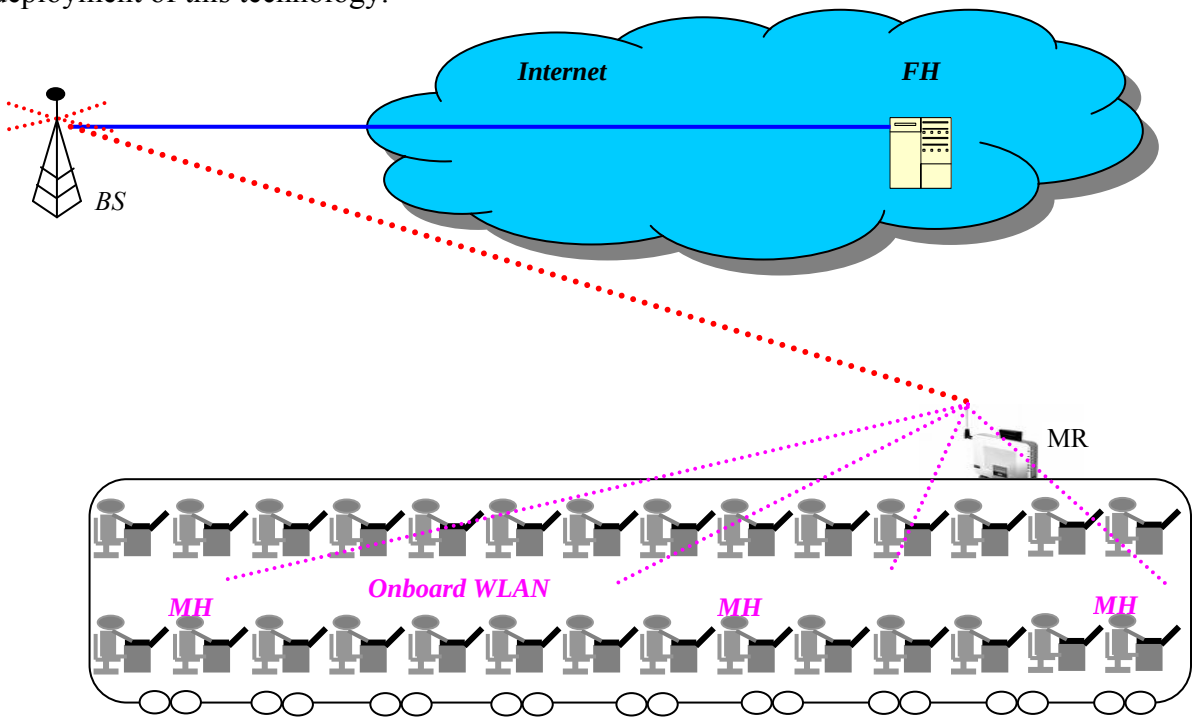

Figure 1: NEMO Connectivity Model

While NEMO offers an attractive solution to fast and reliable Internet access onboard a moving vehicle, it raises several performance issues of its own. These issues (see [3]-[5] for some of the key issues currently being investigated by the research community) arise primarily due to the *existence of multiple wireless links*. Unlike the traditional terminal mobility, where the mobile hosts (MHs) connect to the cellular base station (BS) directly, passengers using the NEMO solution encounter an additional wireless link (MR-MH) before their MHs get connected to the wired infrastructure. Investigating the impact of this additional wireless link on the performance of the widely used TCP protocol, and designing mechanisms to alleviate any negative impacts, is the primary objectives of our study.

The performance problem for TCP in communication environments involving wireless links is well known. Because TCP was originally designed to operate over wired links with negligible loss probabilities, any packet loss is simply treated by TCP as signs of network congestion. Consequently, TCP invokes its elaborate congestion control routines whenever there is a random packet loss on the wireless link, resulting in a very slow recovery for the lost packets. To speed up the recovery process, most classical solutions employ some sort of agents that buffer the passing TCP packets close to the MHs, e.g., in cellular base stations, to perform fast local retransmission in case the packet is lost on the wireless link. These solutions are known to improve the performance of TCP over wireless links dramatically. Although NEMO will benefit from such existing agents, these solutions may not provide the optimum performance enhancement because they would treat the BS-MH link as a single link. Any packet loss in this part of the path, whether they occur in the BS-MR link, or the MR-MH link, will have to be detected and

retransmitted by the agent located at the BS. Clearly, there is a room for further improving TCP performance in the context of NEMO by incorporating the MR into the performance enhancing design.

In this paper, we propose obTCP, a novel performance enhancement scheme for TCP, to address the dual wireless links in NEMO. obTCP uses agents at both BS and MR to quickly recover from wireless losses in BS-MR and MR-MH links. We provide a loss recovery time analysis, which shows that in comparison to snoop, the recovery time improvement increases linearly with delays and exponentially with loss probabilities in the wireless links. These observations are further substantiated by extensive ns-2 simulations, which confirm that recovery time improvements are directly translated to TCP throughput improvements. We further extend our analysis to obtain throughput models for snoop and obTCP. In our simulation experiments, we observed throughput improvement of up to 42% over snoop at 20% packet loss probability. Finally, since network mobility is likely to coexistence with, rather than replacing, the traditional (terminal) mobility, we investigated the fairness issue when obTCP coexists with snoop in the same BS. Using simulations, we demonstrate that the presence of obTCP does not have any serious fairness effect on any coexisting snoop.

The rest of the paper is organized as follows: In Section II we present some notable link layer protocols for terminal mobility, TCP enhancement schemes for NEMO and motivations for this study. The agent functionalities of obTCP are presented in Section III. Section IV presents timing diagrams of loss recovery operation of snoop and obTCP. In Section V, we present loss recovery time analysis followed by numerical experiments. Simulation experiments and the results are discussed in Section VI. In Section VII, we

present the throughput models for snoop and obTCP. We also provide simulation results to validate the throughput models. The coexistence issue of obTCP with other TCP enhancing schemes is explored in Section VII. Section VIII concludes the paper.

II. Related Works

A. Link layer protocols for terminal mobility

Several papers in the literature have proposed link layer protocols as an effective solution for improving TCP performance in terminal mobility [6]-[10]. The protocols try to reduce the effect of high packet loss probability of wireless link in the performance of TCP. The protocols try to recover from wireless losses by locally retransmitting the lost packets. The snoop protocol [6] is one of the major developments in this category.

The snoop protocol [6] is a TCP aware link level protocol. It uses an agent installed at the BS. The role of the agent is to cache TCP packets for each TCP connection. If the agent receives an ACK, it removes the corresponding packets from the cache and forwards the ACK to the FH. However, if a packet is lost in the wireless link, the MH generates DUPACKs. The agent intercepts and drops these DUPACKs, and retransmits the lost packet. Hence, the FH is kept unaware of this wireless loss thereby preventing unnecessary invocation of congestion control mechanisms by the FH. In addition, the agent starts a retransmission timer for each packet it transmits. If the timer expires, the agent retransmits the packet.

Although snoop is quite effective in dealing with wireless losses, it also suffers from performance problem. If multiple losses occur from a window of data, it can recover only one packet per RTT. If the wireless links are slow such that RTT is large enough to cause

the sender timeout that leads to the retransmission at the FH when the retransmission is being performed on the wireless link. Thus, the protocol requires small RTT in the wireless link to allow multiple local retransmissions.

As a result, several proposals ([11]-[13]) have been made in the literature to improve the performance of snoop protocol. But, those proposals are made only for terminal mobility. So. We do not discuss them here.

B. TCP enhancement schemes for NEMO

As of today, there exist only four proposals to enhance TCP performance in NEMO as shown in Figure 2. We categorize the protocols in three groups: wireless loss recovery, connectivity recovery, and fairness. Most of the proposed protocols try to adapt TCP behavior after a handoff. There are three proposals in connectivity recovery category: Freeze TCP model [14], Adaptive packet combining (APC) [15], Store and apply scheme [16]. There is only one proposal in fairness category: MR based fairness control scheme [17]. In general, the proposals in connectivity recovery category try to adapt TCP behavior after a handoff takes place. Freeze TCP eliminates the negative impact of handoff when the handoff takes place between similar networks. On the other hand, APC, store and apply schemes improves TCP performance when vertical handoff takes place. The MR based fairness control scheme guarantees fair share of available bandwidth to the MHs in NEMO. From Figure 2, we find that no attempt has been made to deal with the negative impact caused by dual wireless links of NEMO. Also, the fairness issue in co-existence of terminal mobility and NEMO is not studied yet.

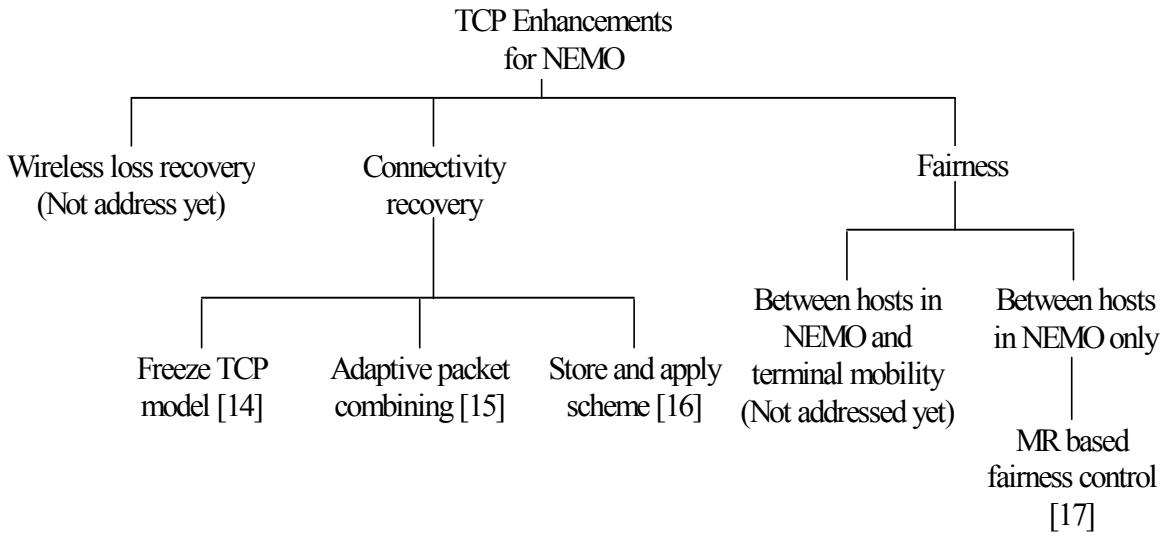

Figure 2: TCP enhancement schemes for NEMO

C. Motivation

TCP performs poorly in wireless networks. TCP invokes its congestion control routines whenever there is a random packet loss on the wireless link, resulting in a very slow recovery for the lost packets. To speed up the recovery process, most classical solutions employ some sort of agents that buffer the passing TCP packets close to the MHs (e.g., in cellular BS) to perform fast local retransmission in case the packet is lost on the wireless link [6]-[13]. These solutions are known to improve TCP performance over wireless links dramatically. Although NEMO will benefit from such existing agents, these solutions may not provide the optimum performance enhancement because they would treat the BS-MH link as a single wireless link, while, in practice, it is a concatenation of two consecutive wireless links.

Unlike the traditional terminal mobility, where the MHs connect to the BS directly, users in NEMO encounter an additional wireless link (MR-MH) before their MHs get

connected to the wired infrastructure. With existing agents, any packet loss in this part of the path, whether they occur in the BS-MR link, or in the MR-MH link, will have to be detected and retransmitted by the agent located at the BS. *So, although the existing agents are able to detect wireless losses, they are unable to locate the origin of wireless losses. As a result, the existing agents may take long time to detect and recover wireless losses. So, the existing agents may not provide optimum performance in NEMO.* Investigating the impact of this additional wireless link on the performance of the widely used TCP protocol, and designing mechanisms to alleviate any negative impacts may solve the problem to some extent. *The objective of this paper is to extend the single point recovery mechanism to multipoint i.e., link-to-link recovery mechanism. In this case, the wireless losses in different wireless links could be recovered independently and simultaneously, thereby decreasing the loss recovery time.*

Also, since NEMO is likely to co-exist with terminal mobility, any TCP enhancement scheme designed for NEMO must share available network resources fairly with the TCP schemes for terminal mobility. To ensure the fairness between the TCP enhancements from these two types of mobility scenario is an essential feature for them to be widely deployed. So, it is quite rewarding to study fairness issues between the TCP enhancement schemes.

III. On-board TCP (obTCP)

In this section, we describe agent functionalities in obTCP for data transfer from FH to MH direction. Figure 3 and Figure 4 summarize these functionalities.

A. obTCP agent at BS

For each connection obTCP keeps track of incoming packet sequence numbers. When a packet arrives, it is stored in the transmission queue for transmission over the wireless link. Once the packet is sent, a copy of it is stored in the cache for possible local recovery later on. The obTCP agent will receive two types of ACK packet from MR: Standard TCP ACK and Selective Negative ACK (SNACK) packet. If an ACK is received it is forwarded to the FH and the buffer spaces are freed. However, if a SNACK is received, it checks its cache. If the packet is found, it is retransmitted immediately over wireless link. Otherwise, it assumes that the packet has been lost due to congestion in wired network or flushed prematurely from the cache. In this case, the obTCP agent sends an indication (congestion packet) to the MR saying not to suppress the Duplicate ACKs (DUPACKs) for these lost packets. Here, the observation is that, if MR suppresses the DUPACKs, it is unnecessarily delaying the Fast Retransmission from the FH because the packets are not available at the BS. In order to activate Fast Retransmission as early as possible, MR should not suppress the DUPACKs for these lost packets.

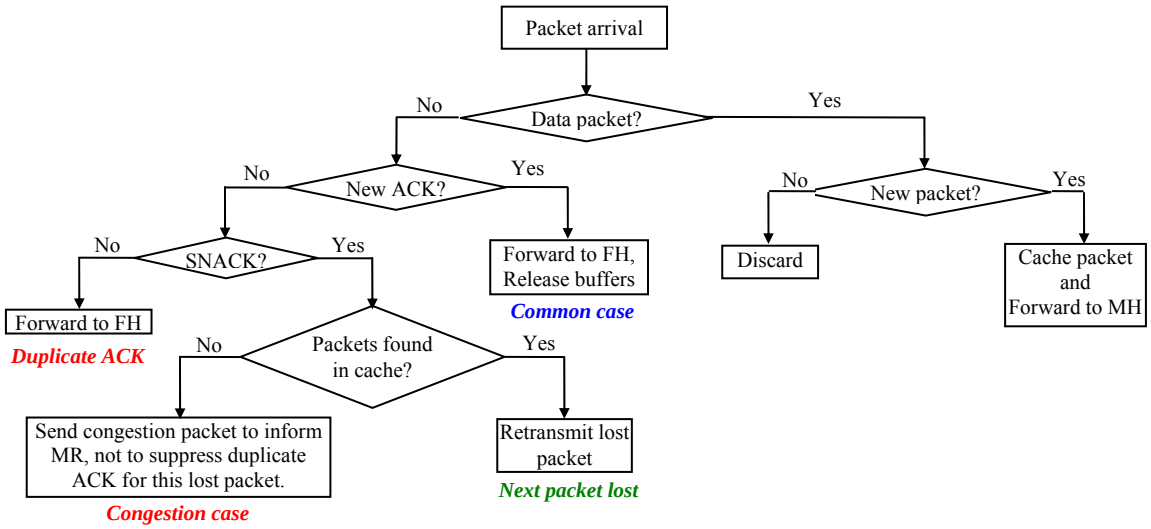

Figure 3: obTCP agent at BS

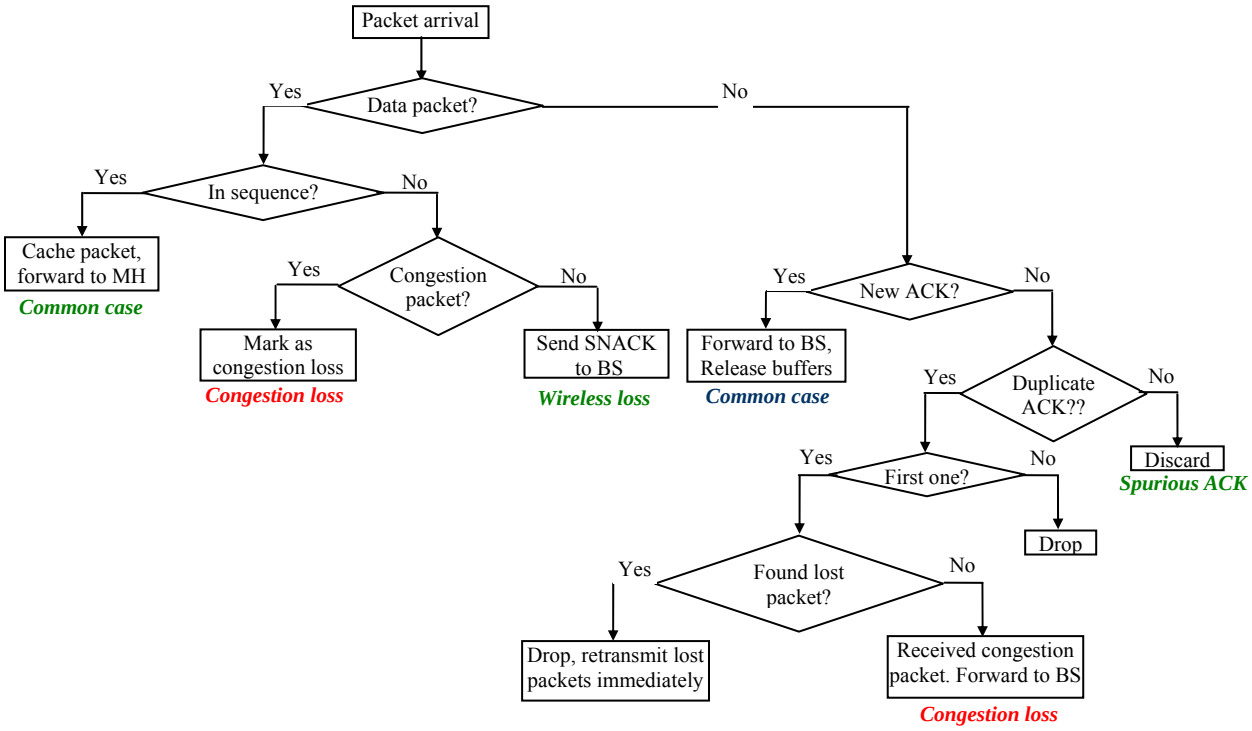

Figure 4: obTCP agent at MR

B. obTCP agent at MR

obTCP agent at MR has four main functions: i) caching TCP packets received from BS, ii) dropping DUPACKs, iii) detecting and reporting packet corruption to the BS, iv) retransmitting packets those are lost in the MR-MH wireless link. If the obTCP agent finds a gap in sequence number of the received packets, it generates a SNACK specifying all the packets those might have been lost in the wireless link and forwards to the BS. If the packets reach the MR in sequence the obTCP agent stores them in the cache and forwards to the MH. The reason behind caching at MR is that the MHs may be connected to the MR via wireless links. When the packets reach the receiver out of order, MH generates DUPACKs. There can be three reasons for which the MH generates these DUPACKs: the packets might have been lost in the path between MR and MH or in the

path between BS and MR, or in the wired network between FH and BS. When DUPACKs reach the MR, the obTCP agent checks its cache. If the packet is found it is retransmitted. Otherwise, it has definitely received an indication (congestion packet) from BS about this packet. If the packet has been lost in wired network, it will get an indication from the BS. In this case, the obTCP agent at MR will not suppress these DUPACKs in order to initiate fast retransmission at the FH.

IV. Comparison of Loss Recovery in snoop and obTCP

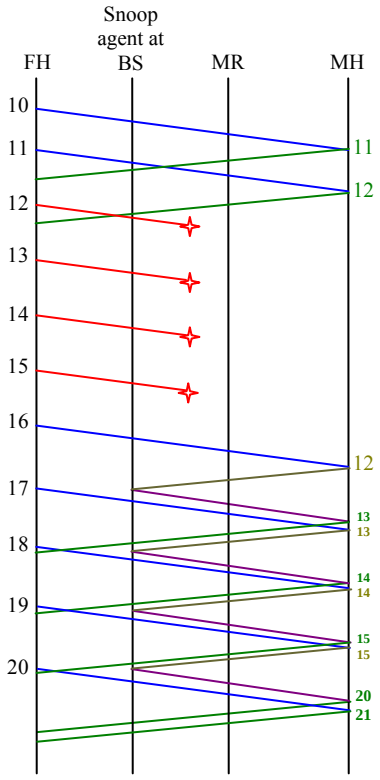

Figure 5: snoop

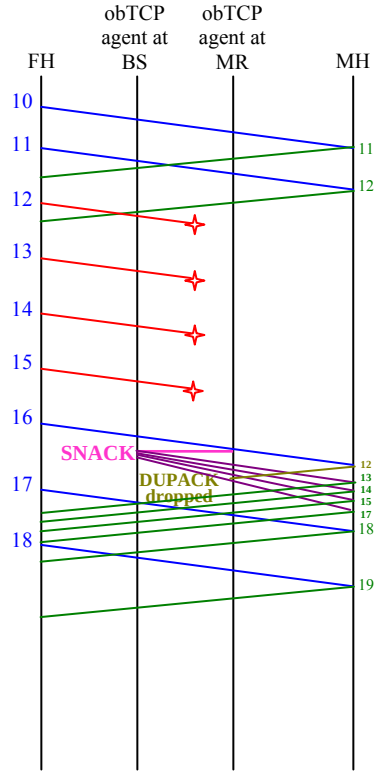

Figure 6: obTCP

Figure 5 and 6 show an example of link layer mechanisms and point out that these mechanisms must be used very carefully. Assume that packets up to sequence number 11 have been transmitted successfully and that packets 12, 13, 14 and 15 have been dropped

in the BS-MR wireless link. Figure 5 depicts how wireless losses are recovered using snoop agents. In this case, the snoop agent treats two wireless links, BS-MR and MR-MH, as a single link BS-MH. When the MH receives packet 16, the MH sends a DUPACK for packet 12. This DUPACK, when received by the snoop agent at BS, makes it retransmit packet 12 from its cache and drop the DUPACK for 12. When the MH receives packet 12 it generates ACK 13. Packet 17 generates DUPACK for packet 13 and snoop agent at BS also drops this. This process continues until all the lost packets are successfully recovered. Therefore, snoop can recover from packet losses in any wireless link but only one packet per RTT over BS-MH wireless link. So, even if snoop is quit effective in dealing with wireless losses, it takes longer time to detect and recover the lost packets as can be seen from the timing diagram. When the packets are lost in MR-MH link, operation of snoop remains same.

In order to rectify the problem of unnecessarily waiting longer for the DUPACK at BS, obTCP includes MR in its design by placing an obTCP agent in MR. In this case, the path from BS to MH consists of two segments: one wireless link between BS and MR, another wireless link between MR and MH. Packet losses in each wireless link are handled separately. So, the wireless losses can be detected at an earlier time than snoop. This is explained in Figure 56 where it can be observed that after receiving the first out-of-order packet at MR, the obTCP agent at MR sends an SNACK packet to BS causing retransmission of all missing packets locally at once, in a lot shorter RTT than if the snoop agent has waited for the DUPACK. Following the same example, assume that packets 12, 13, 14 and 15 are lost. Assuming that all these packets are present in the obTCP cache at BS, MR generates an SNACK for packets 12, 13, 14 and 15 on reception

of packet 16 at MR. On reception of this SNACK packet; the obTCP agent at BS retransmits the requested packets. By using the MR, obTCP helps in reducing the loss recovery time and also enables retransmission of multiple packets in one local (and considerably shorter) RTT thus maintaining a good flow of packets. Note that one could use snoop to recover from multiple losses by introducing the SNACK mechanism at both the BS and the MH. However, that would require changes in the installed base making the deployment of snoop more difficult[*]. On the other hand, obTCP does not require any modification to the existing TCP implementations, yet is capable of exploiting the SNACK mechanism for recovering from multiple losses.

V. Analysis of Loss Recovery Time

Conventional TCP adjusts its congestion window size (w) according to two algorithms, namely slow start and congestion avoidance, where w is inversely proportional to RTT and square root of loss probability (p) [19]. Given that TCP throughput is directly proportional to the window size, we have:

$$w \propto \frac{1}{RTT} \text{ and } w \propto \frac{1}{\sqrt{p}} \tag{1}$$

$$Throughput \propto w \tag{2}$$

As a result, when either delay or loss probability increases, TCP throughput deteriorates significantly. To overcome this, obTCP attempts to reduce the effect of high loss probability in wireless links by quickly recovering from the wireless losses thereby

[*] Even if the MH were to generate the SNACK, the SNACK mechanism would have to work over an RTT spanning the entire BS-MH link, which is much longer than the SNACK RTT in obTCP.

keeping RTT of the connection as low as possible. Hence, the benefit of obTCP depends on *loss recovery time*.

Let us assume that the loss probability in BS-MR and MR-MH links are p_1 and p_2, respectively, and delay in BS-MR and MR-MH links are d_1 and d_2, respectively. In the following subsections, we model the *loss recovery time* and analyze the effectiveness of *link-link loss recovery* mechanism of obTCP. For easy reference, Table 1 lists the variables used in this paper.

A. Modeling the Loss Recovery Time

We know that the recovery process for snoop takes place only at the BS. Therefore, wherever the packet is lost (either BS-MR or MR-MH wireless link), the retransmissions happen from BS only. Hence, the effective loss probability for snoop (p_s) is

$$p_s = 1 - (1 - p_1) * (1 - p_2) \tag{3}$$

Table 1: List of Notations

Notations	*Meaning*
d_1, d_2	*One way delay in BS-MR and MR-MH wireless link respectively*
p_1, p_2	*Loss probability in BS-MR and MR-MH wireless link respectively*
P_s	*Effective loss probability for snoop*
R_s, R_o	*Loss recovery time for snoop and obTCP respectively*
w_s, w_o	*Congestion window size for snoop and obTCP respectively*
T_s, T_o	*Throughput for snoop and obTCP respectively*
R_{gain}	*Gain in recovery time for obTCP over snoop*
N	*Total number of transmissions of a packet over wireless link*

It is easy to see that, for snoop, the total number of transmissions (N) required to successfully receive an ACK at BS is given by:

$$N = \frac{1}{1 - p_s} \tag{4}$$

Hence, the recovery time for snoop is given by:

$$R_S = 2 * (d_1 + d_2) * \frac{1}{1 - p_s} \tag{5}$$

As the packet losses in different wireless links are handled separately in obTCP, we consider them as independent events. Hence, the number of transmissions in each link can be given by replacing p_s in Equation (4) by p_1 for BS-MR link and by p_2 for MR-MH link. Then, the recovery time for obTCP can be given by:

$$R_o = 2 * d_1 * \frac{1}{1 - p_1} + 2 * d_2 * \frac{1}{1 - p_2} \tag{6}$$

The gain (difference) in recovery time is:

$$R_{gain} = R_s - R_o = \frac{2 * d_1 * p_2 + 2 * d_2 * p_1}{(1 - p_1) * (1 - p_2)} \tag{7}$$

Equation (7) is valid for $p_1<1$ and $p_2<1$. Note that, R.H.S. of Equation (7) is always positive which indicates positive gain in recovery time for all loss probability and delay values. From Equation (7), we can conclude that the gain in recovery time increases exponentially with loss probability in both BS-MR and MR-MH links. Small increases in p_1 and p_2 results in higher recovery time gain, reducing the RTT of the connection, which in turn allows w to grow faster (Equation (1)). Also, the gain in recovery time increases linearly with RTT in BS-MR and MR-MH link. Hence, according to Equations (1) and (2), we can expect that end-to-end throughput also increases (i) exponentially with loss

probability in BS-MR and MR-MH link and (ii) linearly with delay in BS-MR and MR-MH link.

As the errors in the wireless links are independent, losses may occur in both links or in any one link. If the losses take place in BS-MR link only, then Equation (7) can be rewritten as:

$$R_{gain} = 2 * d_2 * \frac{p_1}{1 - p_1} \tag{8}$$

If losses occur in MR-MH link only, then Equation (7) can be rewritten as:

$$R_{gain} = 2 * d_1 * \frac{p_2}{1 - p_2} \tag{9}$$

The observations from Equations (8) and (9) can be summarized as:

1. Gain in Recovery time increases exponentially with loss probability of the erroneous link.
2. Gain in Recovery time increases linearly with delay on the error free link.
3. Gain in Recovery time is constant with delay in the erroneous link.

Now, we establish the relationship between window size, throughput and loss recovery time. It is obvious that RTT of a TCP connection is directly proportional to loss recovery time. Using Equations (1), (5), and (6), we get

$$\frac{w_o}{w_s} \propto \frac{R_s}{R_o} \tag{10}$$

Using Equations (1), (2) and (10), we get

$$\frac{T_o}{T_s} \propto \frac{w_o}{w_s} \propto \frac{R_s}{R_o} \tag{11}$$

From Equation (11), it is clear that window size is larger for obTCP, which results in higher throughput for obTCP. This is due to the fact that recovery time in snoop is higher than obTCP. Note that throughput improvement is linearly related with delay and exponentially with loss probabilities as in the case of gain in loss recovery time.

B. Numerical Analysis

We first examine the case when losses are present on both links. Figure 7 and Figure 8 show the performance gain of obTCP over snoop in terms of *gain* in recovery time. For Figure 7, we use p_1=0.15, p_2=0.05, and, for Figure 8, we use d_1=20ms, and d_2=20ms. In Figure 7, we plot R_{gain} as a function of delay in MR-MH link, d_2. From Figure 7, it can be seen that gain increases linearly with delay in both links. In Figure 8, we plot gain in recovery time R_{gain} as a function of loss probability in BS-MR link p_1. From Figure 8, it is evident that gain increases exponentially with loss probability in BS-MR and MR-MH link. Hence, with small increase in loss probability in BS-MR and MR-MH link, obTCP can achieve significantly higher performance than snoop.

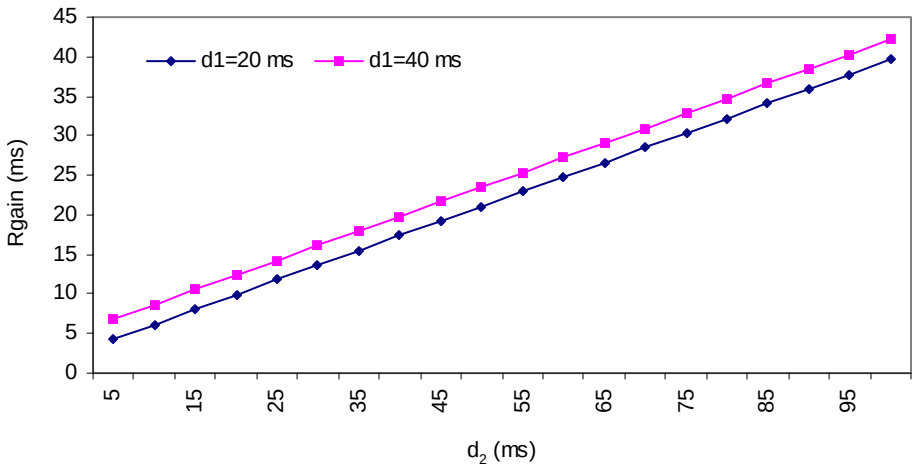

Figure 7: Effect of delay in MR-MH link for losses in both links

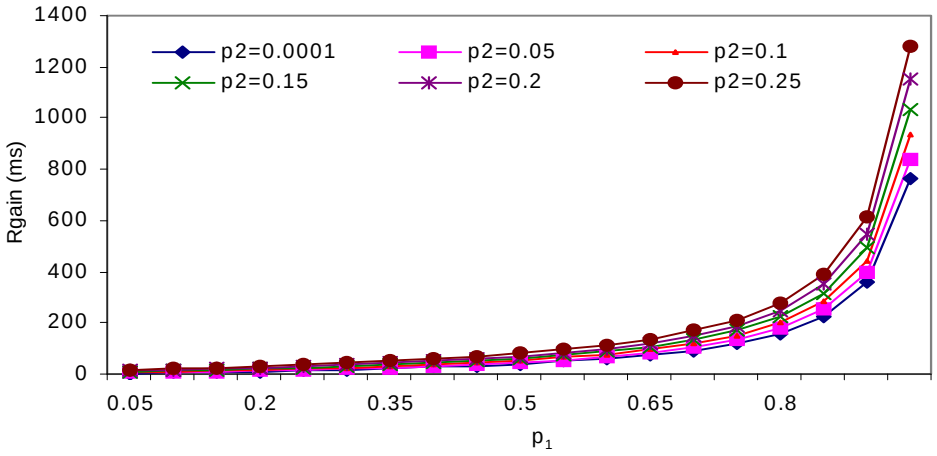

Figure 8: Effect of loss probability in BS-MR link for losses in both links

We now explore the cases when one link is erroneous and the other is error free. For simplicity, we present the results when MR-MH link is erroneous and BS-MR link is error free. In Figure 9 and Figure 10, we see the same trend (as of Figure 7 and Figure 8) for packet losses in MR-MH link. We also study the effect of delay in erroneous link on gain in recovery time. Figure 11 presents gain in recovery time as a function of delay in the erroneous link. We see that gain in recovery time is constant with delay in erroneous link, i.e. delay of erroneous link has no effect on R_{gain}.

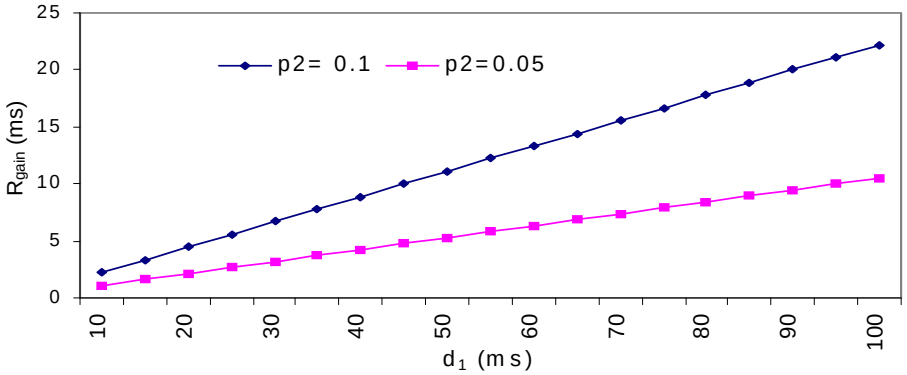

Figure 9: Effect of delay in error free BS-MR link for losses in MR-MH link

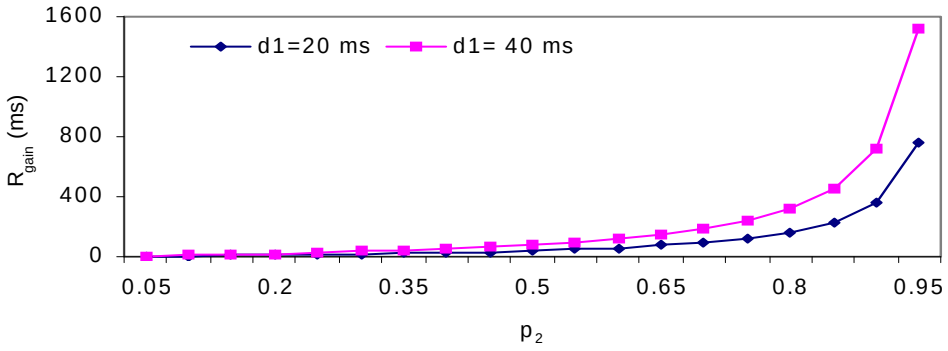

Figure 10: Effect of losses in MR-MH link when BS-MR link is error free

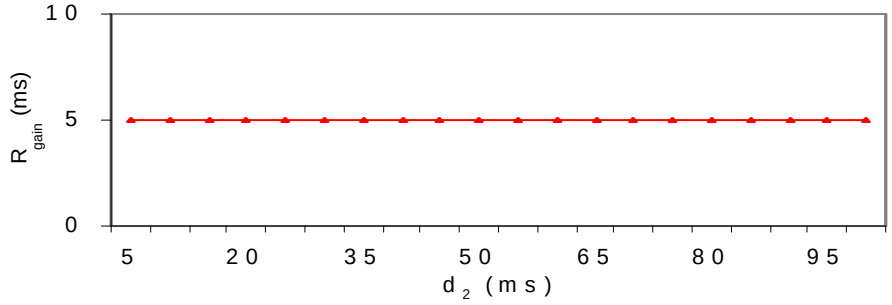

Figure 11: Effect of delay in MR-MH link for losses in MR-MH link

We now study the effect of delay and loss probability on the throughput performance of obTCP and snoop. In Figure 12 and Figure 13, we plot throughput ratio (Equation (11)) as a function of delay and loss probability in BS-MR wireless link respectively. We use the value of one for proportionality constant. For Figure 12, we use p_1=0.1, p_2=0.1, and d_2=10ms. For Figure 13, we use p_2=0.1, d_1=10ms, and d_2=10ms.We see that throughput ratio is constant for all delay values in BS-MR wireless link (Figure 12), which implies that throughput gain is linearly related with delay. We also note that, the throughput ratio is exponentially related with loss probability (Figure 13), which indicates that throughput improvement is exponentially related with loss probability over wireless links.

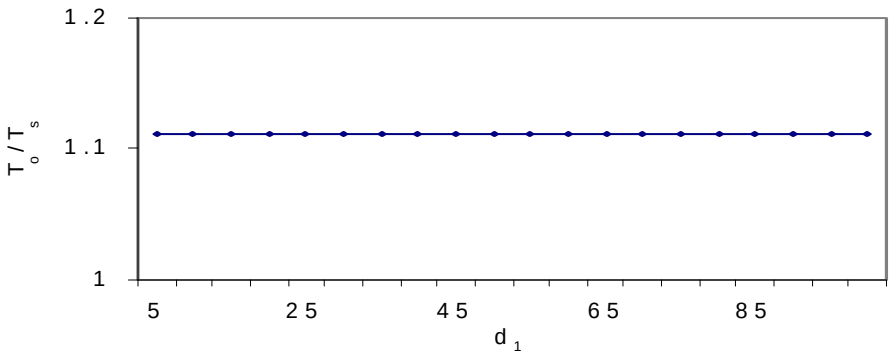

Figure 12: Variation of throughput ratio for delay in BS-MR link

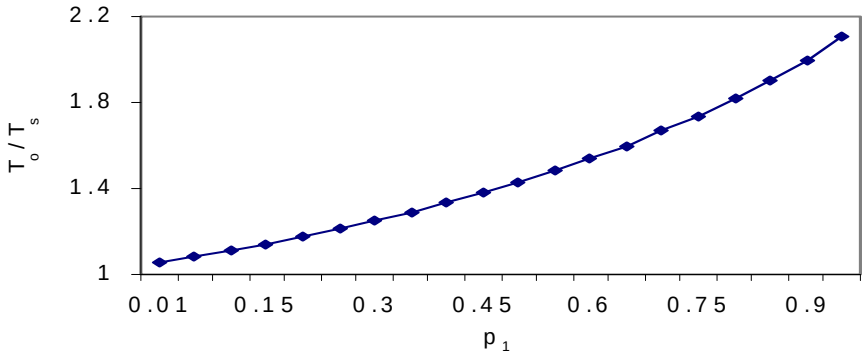

Figure 13: Variation of throughput ratio for delay in BS-MR link

VI. TCP Simulation

To validate our claims, we evaluate throughput performance of obTCP and snoop in network simulator ns-2.29 [20]. Table 2 lists the TCP parameters used in our simulations. We assume a 100 Mbps wired link between FH and BS, a 2 Mbps wireless link between BS and MR (Cellular), an 11 Mbps wireless link between MR and MH (IEEE 802.11b). Wireless link delays are varied from 5ms to 35ms [10], [21]. Loss probability in wireless links is varied from 0.0001 to 0.3 [10], [21], [22]. We choose these values to demonstrate the fact that when wireless link condition deteriorates (both delay and loss probability increases) obTCP performs much better than snoop. It has been observed that, when

channel conditions are good (low delay variation, negligible loss probability), both snoop and obTCP performs similarly with negligible gain in throughput of obTCP. For the sake of clarity of presentation, we present results for three cases only: losses occur in BS-MR link only, wireless links are identical, i.e. loss probability in both links are same, and effect of delay in erroneous link. The results presented in this paper are taken from 2000 sec run of the simulation.

Table 2: TCP Parameters

Parameter	*Value*
TCP version	*Reno*
Packet size	*1000 Bytes*
Initial congestion window	*2 packets*
Maximum congestion window	*16 packets*
Initial slow start threshold	*10 packets*
Minimum retransmission timeout	*0.3 sec*

A. Effect of losses in BS-MR wireless link

Figure 14 and Figure 15 show the throughput performance of obTCP and snoop for d_1=10ms and p_2=0.0001. For Figure 14, we use p_1=0.1, and, for Figure 15, we use p_1=0.2. It is interesting to note that performance of snoop degrades more sharply than obTCP with increasing delay in MR-MH link, which indicates that throughput gain increases with increase in delay of MR-MH link.

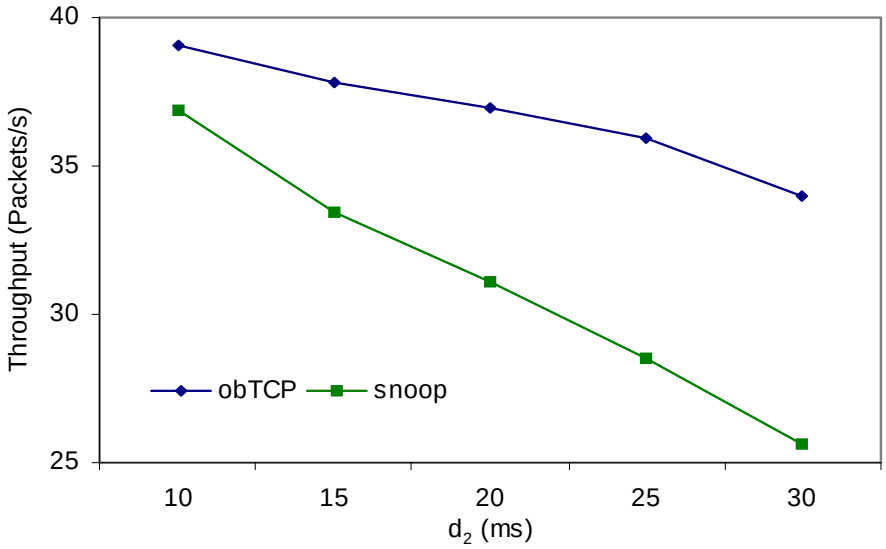

Figure 14: Variation of throughput for loss probability 10% in BS-MR link

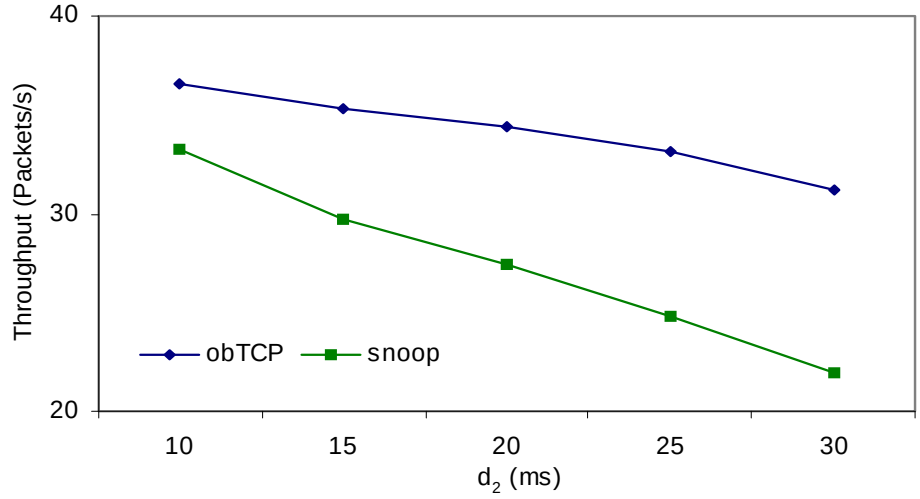

Figure 15: Variation of throughput for loss probability 20% in BS-MR link

A more detailed picture of performance gains of obTCP over snoop are shown in Figure 16-Figure 17. We see that obTCP achieved 33% and 42% throughput gain over

snoop for p_1=0.1 and p_1=0.2 respectively. It can be seen from Figure 16 that with losses in BS-MR link, throughput gain increases linearly with delay in MR-MH link.

We see the exponential increase of throughput improvement from Figure 17 for d_1=20ms, d_2=20ms and negligible loss probability p_2=0.0001. obTCP achieved an improvement of over 39% over snoop.

The higher performance of obTCP over snoop can be explained as follows: When packets are lost in BS-MR wireless link, the obTCP agent at MR detects the loss immediately and requests BS obTCP agent to retransmit the lost packets. So, the recovery mechanism has immediate reaction. But, for snoop, it has to wait for RTT over MR-MH wireless link to even detect the loss. Also, as SNACK mechanism is used over BS-MR wireless link, multiple packet losses are recovered in one RTT over BS-MR wireless link. But, in snoop, only one lost packet is recovered in one RTT over BS-MR and MR-MH wireless link.

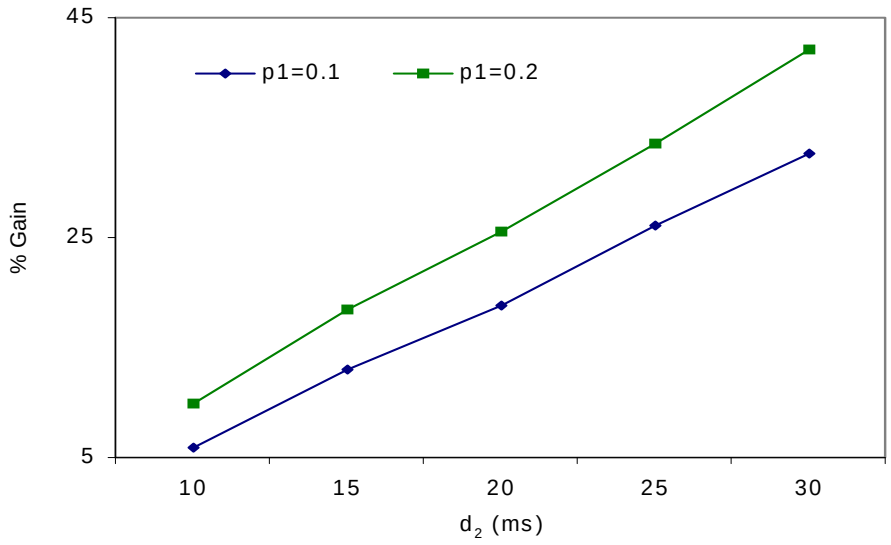

Figure 16: Linear increase in throughput gain for losses in BS-MR link

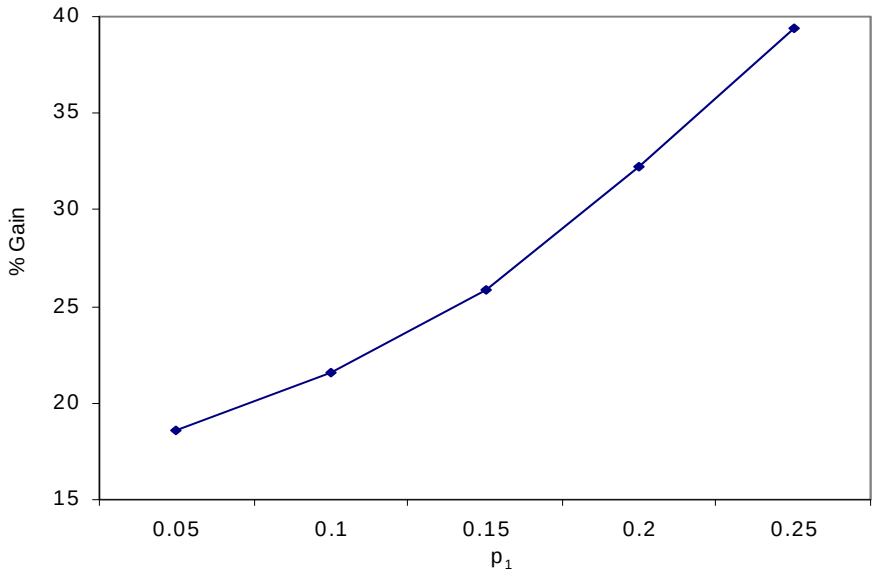

Figure 17: Exponential increase in throughput gain for losses in BS-MR link

B. Effect of losses in identical wireless link

Now, we study the effect of delays in both links, considering that the BS-MR and MR-MH links are identical, i.e. $p_1=p_2$. In both link we use loss probability of 0.05. We vary the delay in both link but each time we keep $d_1=d_2$. Performance results are shown in Figure 18-Figure 19. We see that performance gain increases linearly with delay in both links. In this case, obTCP achieves up to 18% performance gain over snoop.

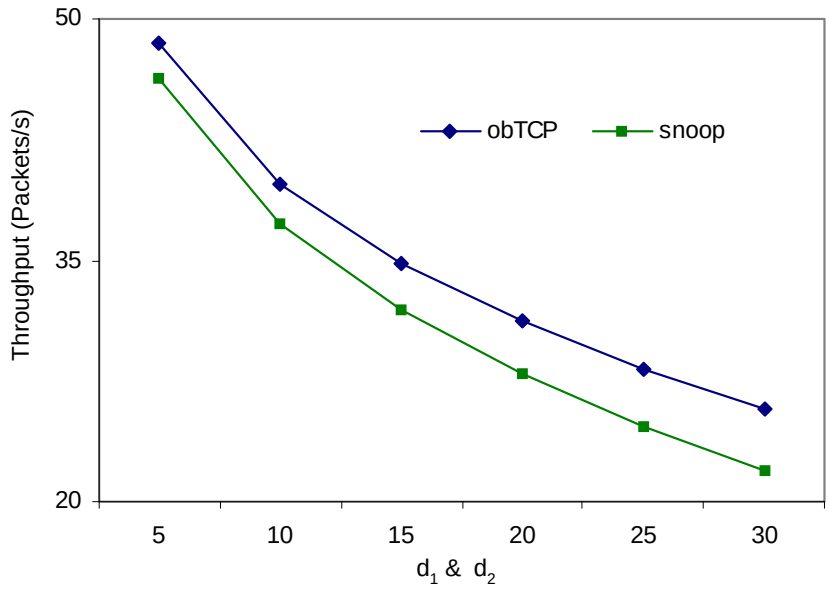

Figure 18: Throughput performance for identical links

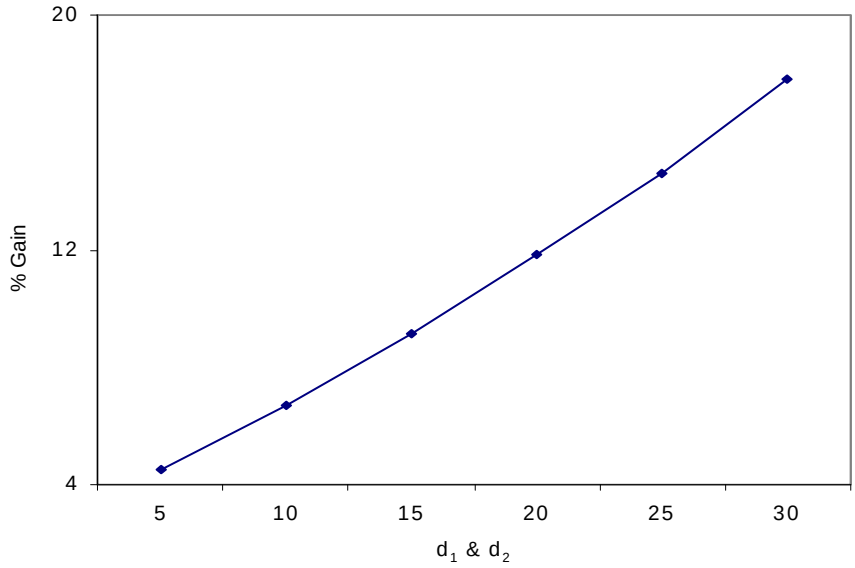

Figure 19: Linear increase in throughput gain for identical links

C. Effect of Delay in erroneous link

In Section V, through numerical analysis, we have shown that gain in recovery time is independent of the delay in erroneous links. We conduct several simulations to verify our claims. Results are shown in Figure 20-Figure 21. For Figure 20 we use p_1=0.1, p_2=0.0, d_2=20ms, and for Figure 21 we use p_1=0.0, p_2=0.1, d_1=20ms. Figures show that our claim matches for simulation experiments too. In this case, when the delay is increased in erroneous links, both protocols are affected by this increased delay. So, the performance gain depends only on the delay on the error free link, which is kept constant. Hence, the performance gain becomes constant and the absolute value of gain depends on the delay of the error free link.

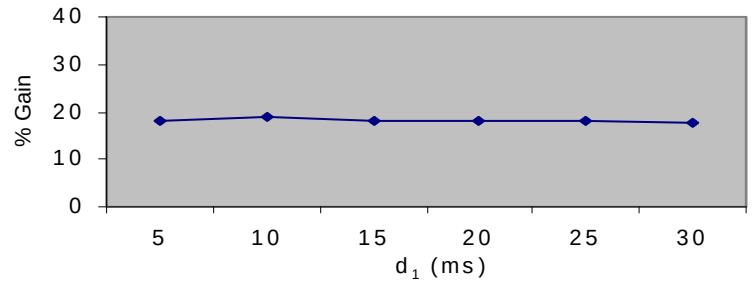

Figure 20: Constant gain for delay in erroneous link BS-MR

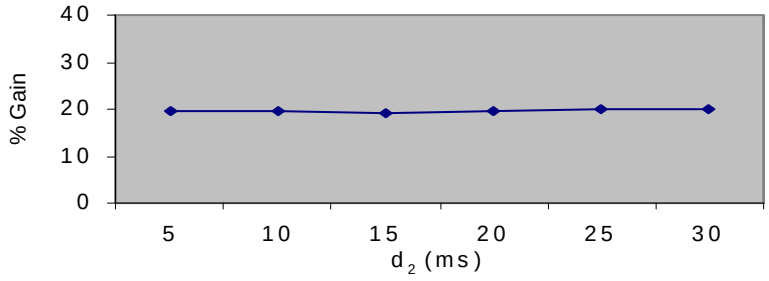

Figure 21: Constant gain for delay in erroneous link MR-MH

F. RTT seen by TCP Reno

In this section, we describe some notable reasons for which obTCP achieved better performance than snoop in all simulations described above. We study how quicker recovery from wireless losses helps TCP to keep RTT of the connection as low as possible, which, in turn, helps in faster growth of congestion window. We use d_1=10ms, d_2=10ms, p_1=0.1 and p_2=0.0001. Figure 22 shows the Smoothed RTT (SRTT) of the connection measured between t_1=1000s and t_2=1050s of the simulation. We see that the RTT of the connection is low most of the time for obTCP, which helps in faster growth of TCP congestion window. For example, consider that a packet is lost in MR-MH wireless link. In case of snoop, the loss detection and its possible retransmission will take one RTT spanning BS-MR and MR-MH links. However, in case of obTCP, the agent at MR detects the loss and retransmits the lost packet quickly, which takes one RTT spanning MR-MH link only. As a result, the ACKs for the lost and recovered packets reach the sender much faster, which results in release of new packets faster in obTCP than snoop. Hence, obTCP achieved better performance than snoop.

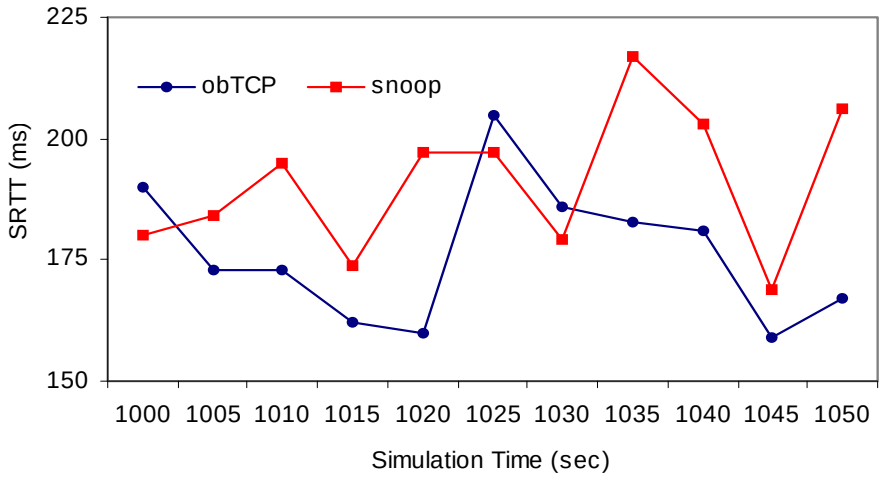

Figure 22: Comparison of smoothed RTT

VII. Throughput Models

We now develop the throughput models for snoop and obTCP in NEMO. We take a similar approach presented in [23], i.e. the window behavior is modeled in terms of rounds. To derive the duration of each round, we use the loss recovery time analysis presented in Section V.

A. Snoop

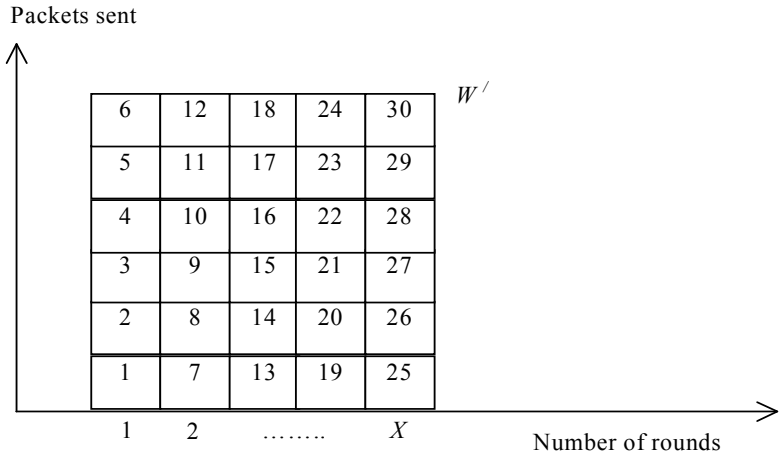

Figure 23: Congestion window evolution for snoop

One important assumption behind the design of snoop is that all wireless losses are recovered using local retransmission mechanism. So, the CWND will reach the maximum size *($W^{/}$)* and stays there for the rest of the communication.

A sample path of the evolution of CWND is shown in Figure 23. In each round, $W^{/}$ packets are transmitted. So, the expected number of packets (*E[Y]*), transmitted in X rounds, can be given as follows:

$$E[Y] = X * W^{/} \tag{12}$$

To derive mean duration of X rounds, let us consider the events of any one round. In a particular round, the number of packet losses *(L_t)* can be given by:

$$L_t = W^{/} * p_s \tag{13}$$

where p_s is given by Equation (3).

Each lost packet is recovered separately. So, the total recovery time is $L_t * R_s$, where R_s is the loss recovery time for each packet and is given by Equation (5). Hence, the end-to-end RTT seen by TCP Reno (t) is given as:

$$t = 2d_0 + 2d_1 + 2d_2 + L_t * R_s \tag{14}$$

So, the mean duration of X rounds, $E[A]$, is:

$$E[A] = X * t \tag{15}$$

The end-to-end throughput, for snoop, can be derived as in [23]:

$$\eta_s = \frac{E[Y]}{E[A]} = \frac{W^{/}}{2d_0 + 2d_1 + 2d_2 + L_t * R_s} = \frac{W^{/}}{2d_0 + 2d_1 + 2d_2 + W^{/} * p_s * R_s} \tag{16}$$

B. ObTCP

To derive throughput expression for obTCP we take similar approach as presented for snoop. The window evolution is the same as shown in Figure 23. The mean number of packets transmitted is given by Equation (12). In a particular round, total number of packet losses in BS-MR link (L_{t1}) is:

$$L_{t1} = W^{/} * p_1 \tag{17}$$

Let us assume that n_1 packets are recovered per SNACK packet. So, the number of SNACK packets generated is $\frac{L_{t1}}{n_1}$. Hence, the total loss recovery time in BS-MR link is $\frac{L_{t1}}{n_1} * \frac{2d_1}{1 - p_1}$, the second term represents the loss recovery time per SNACK packet (first term of Equation (6)).

Similarly, we can obtain the loss recovery time in MR-MH link. The total loss recovery time in MR-MH link is $\frac{L_{t2}}{n_2} * \frac{2d_2}{1-p_2}$, where L_{t2} is the total number of packet losses in MR-MH link, and n_2 is the number of lost packets retransmitted. We note that retransmission of packets from MR follows go-back-*n* ARQ technique.

Hence, the end-to-end RTT, seen by TCP Reno is given as:

$$t = 2d_0 + 2d_1 + 2d_2 + \frac{L_{t1}}{n_1} * \frac{2d_1}{1-p_1} + \frac{L_{t2}}{n_2} * \frac{2d_2}{1-p_2} \tag{18}$$

The mean duration of X rounds, $E[A]$, is:

$$E[A] = X * t \tag{19}$$

where t is given by Equation (18).

Finally, the end-to-end throughput is given by modifying Equation (16) as follows:

$$\eta_o = \frac{W'}{2d_0 + 2d_1 + 2d_2 + \frac{L_{t1}}{n1} * \frac{2d_1}{1-p_1} + \frac{L_{t2}}{n_2} * \frac{2d_2}{1-p_2}} \tag{20}$$

C. Model Validation

In this Section, we describe numerical results for the throughput model of snoop and obTCP (Equation (16) and Equation (20)). We also provide simulation result for obTCP to validate the proposed models. The wired network is assumed to be error free. The delay in wired network from FH to BS is 50 ms, in wireless link between BS and MR is 20 ms, in wireless link between MR and MH is 10 ms. We use a fixed loss probability 0.1% in MR-MH link and vary the loss probability in BS-MR link from 0.1% to 10%. Table 3 shows the TCP parameters used in our simulations. To validate the proposed

models with simulation results, we assume 5 ms MAC delay at BS and MR for every packet transmission. The results are shown in Figure 24.

Table 3: TCP parameters for throughput model validation

Parameter	Value
TCP Version	TCP Reno
Packet Size	1024 Bytes
Initial Congestion Window	2 Packets
Maximum Congestion Window	16 Packets
Initial Slow Start Threshold	12 Packets

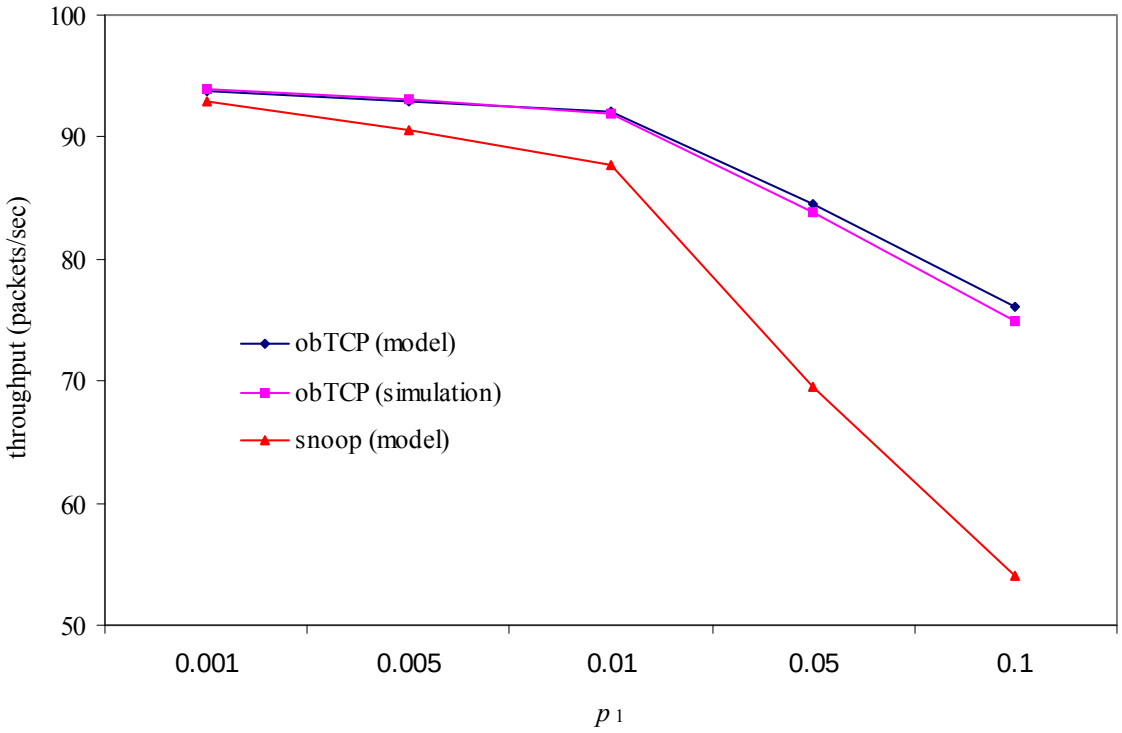

Figure 24: Throughput model validation for obTCP and snoop

From Figure 24, we see that obTCP achieves higher throughput than snoop for all loss probabilities. The performance increase of obTCP over snoop in this case ranges from 1% to 40%. When the loss probability in BS-MR link is 0.001, both protocols perform almost equally. But, as the condition of BS-MR link deteriorates (i.e., loss probability increases), the performance of snoop drops more sharply than obTCP. At loss probability 10%, obTCP achieves 40% throughput improvement over snoop. The higher performance of obTCP over snoop is due to the faster loss recovery mechanism of obTCP than snoop. Interestingly, the simulation result matches exactly with numerical results, thereby validating our models, i.e. our models can accurately predict end-to-end performance in NEMO.

VIII. Co-existence problem

It is established that obTCP achieves considerably better performance than snoop in network mobility. In future, two kinds of wireless mobility are likely to co-exist: (i) terminal mobility, and (ii) network mobility. Both the networks will use the same cellular BS to connect to the Internet. Also, they will use TCP as transport protocol. But it is already established that since TCP is not fit for networks with wireless links [8], we need its enhancements, such as snoop [6], which is a promising candidate for terminal mobility, and obTCP, which is a kind of dual agent solution (unlike single agent solution of snoop) for network mobility. This raises the issue of fairness between obTCP and snoop. Fairness measures the distribution of network resources among the sources using different protocols, such as obTCP and snoop. To assess the extent of fairness issue, we conduct several simulations where both obTCP and snoop co-exist in the same BS. We

use the fairness index function of [24] to quantify the fairness between obTCP and snoop. The fairness index function is expressed as:

$$\omega = \frac{\left(\sum_{i=1}^{n} \lambda_i\right)^2}{n * \left(\sum_{i=1}^{n} \lambda_i^2\right)} \tag{21}$$

where n is the number of flows (i.e., sources in the network) through the bottleneck link, and λ_i is the fraction of the bottleneck link bandwidth obtained by flow i. The value of fairness obtained through this method ranges from $(1/n)$ (i.e., extremely unfair) to 1 (perfectly fair), with 1 indicating equal allocation to all sources.

Link utilization ψ of the bottleneck link is calculated as [24]:

$$\psi = \frac{\sum_{i=1}^{n} \lambda_i}{b} \times 100 \tag{22}$$

where b is the bottleneck link bandwidth.

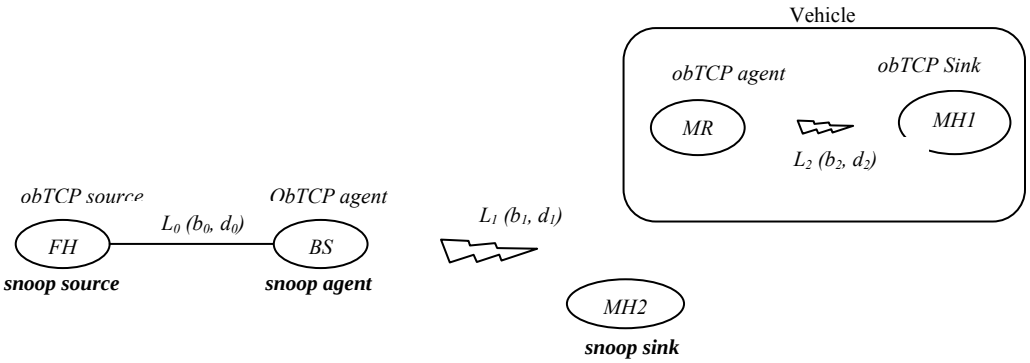

Figure 25: Simulation model for co-existence scenario

Table 4: Bandwidths and Delays of Links

Links	L_0	L_1	L_2
Bandwidth in Mbps	100	2	11
Delay in ms	50	20	5

The network topology is shown in Figure 25. For each link the tuple (b*,d*) indicates the bandwidth and delay of that link. The bandwidth and delay values are summarized in Table 4. We use negligible loss probability in L_2 (p_2=0.0001). Results are summarized in Table 5. The throughput of obTCP and snoop are represented by λ_1 and λ_2 respectively. We see that the network is always nearly evenly used. So, the degree of unfairness in the network is low. We also note that the link utilization is low.

Table 5: Fairness comparison (Low utilization)

p_1	λ_1	λ_2	ω	ψ
0.0	96.3	102.7	0.999	79.6
0.005	78.8	85.7	0.998	65.86
0.01	61.3	65.6	0.998	58.19
0.05	55.7	47.62	0.981	33.64
0.1	49.4	27.17	0.92	21.34

Table 6: Fairness comparison (High utilization)

p_1	λ_1	λ_2	ω	ψ
0.0	12.0	12.03	0.999	96.12
0.005	10.03	9.99	0.999	80.11
0.01	9.5	8.45	0.997	71.8
0.05	7.387	6.327	0.994	54.88
0.1	7.38	5.037	0.966	49.68

To observe if there is any difference in fairness in the case when bottleneck link utilization is high, we conduct a separate simulation in which we use a very low value for the bottleneck link. The same topology is used with only difference of 0.2 Mbps bottleneck link. Table 6 shows the fairness index and link utilization. The results show that shared link utilization is improved considerably. It is also observed that the fairness index is high enough indicating fair share of bottleneck link bandwidth. Hence, obTCP and snoop can co-exist and use the same cellular BS to connect to the Internet.

IX. Conclusion

In this paper, we have proposed on-board TCP (obTCP) to effectively address the wireless link related issues in network mobility. obTCP places agents at both BS and MR to quickly recover from wireless losses. We have presented loss recovery time analysis that shows positive gain in wireless loss recovery time of obTCP over snoop for all loss probability and delay values. We then used the loss recovery analysis to obtain throughput performance of snoop and obTCP. The throughput models are validated through ns-2 simulations. We have shown that gain in recovery time increases linearly with delay and exponentially with loss probability in wireless links. It is also shown that obTCP keeps RTT of the TCP connection low compared to snoop which helps in faster growth of congestion window. Higher congestion window allows obTCP to achieve higher throughput than snoop, e.g. at 25% loss probability obTCP achieves throughput improvement of 52% over snoop.

From the evaluation of obTCP through ns-2 simulation, the following points are proved. First, throughput improvement increases exponentially with loss probability in

erroneous link. Second, throughput improvement increases linearly with delay on the error free link. Finally, throughput improvement is constant with delay in the erroneous link.

Since network mobility is likely to co-exist with conventional terminal mobility we have also examined the fairness issue when obTCP and snoop share the same cellular base station. It is shown that obTCP does not create any negative impact on snoop in the same cellular base station.

References

[1] Icomera-Wireless Onboard Internet: http://www.icomera.com

[2] V. Devarapalli, R. Wakikawa, A. Petrescu, and P. Thubert, *"Network Mobility (NEMO) Basic Support Protocol"*, RFC 3963, January 2005.

[3] A. Baig, L. Libman, and M. Hassan, *"Performance Enhancement of On-Board Communication Networks using Outage Prediction"*, IEEE Journal on Selected Areas in Communications, Vol. 24, issue 9, pp. 1692-1701, 2006.

[4] H. Petander, E. Perera, K. C. Lan, and A. Seneviratne, *"Measuring and Improving the Performance of Network Mobility Management in IPv6 Networks"*, IEEE Journal on Selected Areas in Communications, vol. 24, Issue 9, pp. 1671-1681, 2006.

[5] I. Chan, A. Chunq, M. Hassan, K. Lan, and L. Libman, *"Understanding the Effect of Environmental Factors on Link Quality for On-board Communications"*, IEEE VTC, vol. 3, pp. 1877-1881, September 2005.

[6] Hari Balakrishnan, Srinivasan Seshan, Randy H. Katz, *"Improving Reliable Transport and Handoff Performance in Cellular Networks"*, ACM Wireless Networks, vol. 1, issue 4, pp. 469-481, 1995.

[7] H. Balakrishnan, V. Padmanabhan, S. Seshan, and R. Katz, *"A Comparison of Mechanisms for Improving TCP Performance over Wireless Links"*, ACM/IEEE Transactions on Networking, vol. 5, issue 6, pp. 756-769, 1997.

[8] B. Sardar, and D. Saha, *"A Survey of TCP Enhancements for Last Hop Wireless Networks"*, IEEE Communication Surveys and Tutorials, vol. 8, issue 3, pp. 20-34, 2006.

[9] Fanglei Sun , Victor O.K. Li Soung, C. Liew, *"Design of SNACK Mechanism for Wireless TCP with New Snoop"*, IEEE Wireless Communications and Networking Conference, vol. 5, issue 1, pp. 1046-1051, 2004.

[10] N. Vaidya and M. Mehta, *"Delayed Duplicate Acknowledgements: A TCP Unaware Approach to Improve Performance of TCP over Wireless"*, Texas A&M University, Technical Report 99-033, February 1999.

[11] S. Lee, S. H. Lee, J. S. Lim, *"Fast snoop scheme for TCP connections in wired-wireless environments"*, IEICE trans. comm., Vol. E91-B, No. 9, pp. 2998-2999, September 2008.

[12] J. Kim, K. Chung, *"C-snoop: Cross layer approach to improving TCP performance over wired and wireless networks"*, IJCSNS, Vol. 7, No. 3, pp. 131-137, March 2007.

[13] S. Vandarkar, A. L. Reddy, N. Vaidya, *"TCP-DCR: A novel protocol for tolerating wireless channel errors"*, IEEE trans. mobile computing, Vol. 4, No. 5, pp. 517-529, October 2005.

[14] A. Baig, L. Libman, and M. Hassan, *"Performance Enhancement of On-Board Communication Networks using Outage Prediction"*, IEEE JSAC, vol. 24, issue 9, pp. 1692-1701, September 2006.

[15] N. Hirokazu, M. Wetterwald, and C. Bonnet, *"Adaptive Packet Combining for IPv6 Soft Handover Applied to Network Mobility"*, IEEE PIMRC, pp. 1-5, September 2008.

[16] K. Ishibashi, N. Morishima, and H. Sunahara, *"A Scheme for Adaptive TCP to Drastic Changes in Link Characteristics"*, UBICOMM, pp. 183-188, November 2007.

[17] A. Baig, L. Libman, and M. Hassan, *"Fairness Control by Mobile Routers On-Board Communication Networks"*, IEEE VTC, pp. 753-757, April 2007.

[18] T.Goff, J. Moronski, D.S. Phatak, and V. Gupta, *"Freeze-TCP: A True end-to-end TCP Enhancement Mechanism for Mobile Environments"*, IEEE INFOCOM, vol. 3, pp. 1537-1545, 2000

[19] M. Hassan and R. Jain, *"High Performance TCP/IP Networking: Concepts, Issues, and Solutions"*, Prentice Hall, 2003.

[20] The Network Simulator NS-2: www.isi.edu/nsnam/ns

[21] C. Zheng and V. Tsaoussidis, *"TCP Real: Improving real-time capabilities of TCP over heterogeneous networks"*, IEEE/ACM NOSSDAV, pp. 189-198, 2001.

[22] M. Gerala, M. Sanididi, R. Wang, A. Zanella, C. Casetti, and S. Masco, *"TCP Westwood: Window control using bandwidth estimation"*, IEEE GLOBECOM, Vol. 3, pp. 1698-1702, 2001.

[23] J. Padhye, V. Firoiu, D. Towsley, and J. F. Kurose, *"Modeling TCP Reno Performance: A Simple Model and Its Empirical Validation"*, IEEE/ACM Transaction on Networking, vol. 8, issue 2, pp. 133-145 April 2000.

[24] D. Chiu and R. Jain, *"Analysis of the increase and decrease algorithms for congestion avoidance in computer networks"*, Computer Networks and ISDN Systems, vol. 17, issue 1, pp. 1-14, Sep-Oct, 1989.