To Drop or Not to Drop: On the Impact of Handovers on TCP Performance

Reuven Cohen and Anna Levin
Dept. of Computer Science
Technion, Israel

***Abstract*—This paper presents a comparison between two handover schemes: *drop* and *forward*. In the *drop* scheme, packets received by the base station after the host has disconnected are dropped, whereas in the *forward* scheme these packets are forwarded to the new base station. We analyze various TCP flavors and compare our findings to simulation results. Our results can be used to determine which handover scheme and which TCP flavor should be employed to minimize the negative effect of handovers on TCP performance.**

I. Introduction

Traditionally, cellular networks were based on the circuit switching technology, because they were not intended to support data applications. Nowadays, however, data applications acquire a lion's share of the network bandwidth. In recent years, several approaches have been developed for efficient execution of data applications by cellular users [1], [2]. With all the advantages of running a cellular network using packet switching, the environment of mobile networks is quite different from that of wired networks, for which packet switching technology has been optimized in the last 30 years. The higher loss rates, lower link capacities, and frequent handovers present additional challenges to data protocols.

Data applications such as HTTP require reliable data delivery over the network. TCP is the most widely used transport protocol for this purpose. It was developed to work with fixed networks and optimized to allow good bandwidth utilization with congestion control. However, in the case of handovers, TCP considers packet losses as a sign of congestion. This results in unnecessary timeouts and retransmissions.

This paper presents a mathematical analysis of the TCP throughput for mobile hosts. As explained later, such an analysis cannot simply consider all the outstanding packets during handover as lost, and therefore cannot be performed by extending previous studies for static TCP connections, like [3], [4], [5]. We distinguish between three possible schemes for treating TCP packets that reach a base station after the mobile host has moved to a new cell served by a different base station:

1) *Drop*: the packets are silently dropped by the old base station.
2) *Forward*: the packets are forwarded to the new base station and then to the mobile host; since new packets are routed from the sender to the host on the most direct path, some of the forwarded packets might be received out-of-order.
3) *Forward-and-rearrange*: the packets are forwarded to the new base station and then to the host before new packets that are routed over the direct path; this requires the new base station to delay new packets until the old ones have been forwarded to the mobile host.

We discuss and systematically analyze the impact of handover on various TCP flavors. This analysis takes into account many parameters that affect TCP throughput for mobile hosts:

- The handover rate. Frequent handovers may result not only in time wasted on recovery, but also in *cwnd* limitation, because there may not be enough time between consecutive handovers for the connection to reach its TCP fair share of the available bandwidth.
- The delay on the path between the first common parent and the old base station. This delay defines which portion of the in-flight packets will be dropped by the *drop* scheme or forwarded by the *forward* and *forward-and-rearrange* schemes.
- The time it takes for the forwarded packets to reach the new base station.

The rest of the paper is organized as follows. In Sec. II we present related work. In Sec. III we describe the various handover schemes in greater detail. In Sec. IV and Sec. V we analyze the influence of the *drop* and *forward* schemes on the TCP throughput of an individual connection. Sec. VI presents simulation results and compares them to the results computed using our prediction equations. Sec. VII concludes the paper.

II. Related Work

There are many simulation-based investigations of the problematic aspects of TCP in wireless network environments. For example, [6] shows that long sudden delays, mostly attributed to handovers, are common in the GPRS wireless WAN. It explores the influence of these delays on TCP performance and concludes that the spurious timeouts they trigger may lead to unnecessary retransmissions. Ref. [7] presents a simulation-based investigation for measuring the effect of handovers on several TCP versions. The authors suggest that forwarding might improve TCP performance at handover, pointing out that duplicate packets should be filtered at the access point.

There are also many surveys exploring the problematic aspects of TCP over wireless networks (e.g. [8], [9]). Some of them analyze existent solutions for dealing with TCP throughput degradation due to handovers and lossy links. In [8] the authors conclude that selective acknowledgements and explicit loss notifications can significantly improve performance.

In this paper we develop a model for predicting TCP throughput for mobile hosts. We address three TCP flavors: Reno [10], New-Reno [11] and SACK [12]. Several works propose models for TCP throughput prediction, e.g., [3], [4],

[5], [13], [14]. An overview of these works can be found in [3], [4]. In what follows we focus on the model proposed in [5]. However, the discussion pertains to most of the other models as well.

The authors of [5] present a model for studying the effect of general packet loss caused by network congestion on TCP Reno. It predicts the throughput of a TCP connection as a function of loss rate and RTT. It assumes that if a packet is lost, all the remaining packets transmitted until the end of that round will be lost as well. The effect of handovers on TCP under the *drop* scheme can be studied using this model by considering the loss sequence caused by each handover exactly like the loss sequence caused by congestion. However, the model we use is different for several reasons:

- When some of the packets are dropped due to handover, we do not necessarily assume that all in-flight packets are also lost. If the network devices (routers or switches) in the vicinity of the new and the old BSs are quickly informed about the new location of the mobile node, only a fraction of these packets will be affected by the handover.
- We address also the *forward* scheme. In this case the mobility of the host to a new cell does not necessarily result in loss of packets, but only in out-of-order delivery, which has a different effect on throughput.
- We take into account some network-related parameters that affect the throughput in the case of handover but do not appear in the model of [5].

Another major difference between our model and that of [5] is that we do not address the effect of network congestion on the connection throughput. We do not ignore this effect, of course, but rather use the connection throughput without handovers as a parameter in our equation. This follows from our motivation, which is to understand the effect of various handover schemes on the throughput for different TCP flavors, and not the correlation between packet loss rate and throughput.

In the last years, the impact of mobility on the performance of TCP has drawn a lot of attention. For example, the authors of [15] and [16] perform analysis of the *drop* scheme. In [15], they analyze possible packet drop scenarios in a cellular environment, including handovers, poor wireless link conditions and congestion in the wired network. They show that their analytical results outperform throughput prediction based on the Amherst model. In [16], the authors calculate the probabilities for packet loss, taking into account network congestion and the loss caused by handovers. Then, they incorporate these probabilities into the prediction equations from the Amherst model. The main difference between[15], [16] and our work is as follows. In [15], [16], the authors concentrate on calculating loss probabilities and finding the throughput as a function of the amount of lost data. In contrast, our analysis aims to predict the amount of lost data as a function of the handover frequency and network delays. Another difference is that in [15], [16], only TCP Reno is analyzed. We also address New-Reno and SACK, and show significant differences in their behavior.

In [17], the authors analyze the *forward* scheme for New-Reno. They suggest improvement to the buffer management algorithm in order to prevent overflow, and show that it is better to drop new packets rather than old ones. Our paper presents throughput prediction for TCP Reno and SACK, in addition to analyzing the New-Reno's behavior. Furthermore, our analysis takes into account network parameters, like the delay on the path between the first common parent and the old base station.

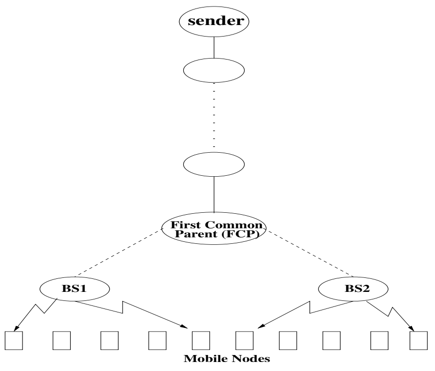


Fig. 1. Network model

In [18], the authors present an overview of mobility management protocols based on Mobile IP, TCP Migrate, and SIP, and study the effect of handover. Their estimation is based on latency and throughput degradation time. Unlike the study in [18], our study is not technology dependent. Our performance estimation is based on the measurement of overall throughput and extra bandwidth requirements. In addition we analyze the *drop* scheme, which is not mentioned among the possible schemes for TCP-based applications in [18].

III. Handover Schemes

A. Drop vs. Forward

Fig. 1 depicts our schematic network model. The sender node represents the server with which the considered host communicates. BS1 is the old base station and BS2 is the new one. The First Common Parent (FCP) is the lowest common predecessor of BS1 and BS2. Throughout the paper we consider the following handover model. When the mobile node decides to switch from BS1 to BS2, it informs both nodes. A control message is then sent by BS1 or BS2 to FCP, asking FCP to direct new packets for this mobile node through BS2 rather than through BS1. We assume that when FCP makes the required change in its forwarding/routing table, the mobile node switches from BS1 to BS2. This is considered the handover time.

The three considered handover schemes – *drop*, *forward,* and *forward-and-rearrange* – are different in their implementation complexity and their effectiveness. At first glance the *forward-and-rearrange* scheme seems to be the most efficient in terms of bandwidth utilization and delivery time minimization. However, our simulation study, presented in Sec. VI, reveals that *forward-and-rearrange* has only a very small advantage over *forward,* and sometimes even compares negatively to it. Therefore, it cannot justify the added complexity. For this reason, and due to lack of space, we focus in the rest of the paper only on *forward* and *drop*.

There are many TCP variants and flavors, the most important of which are TCP SACK[12], Reno[10] and New-Reno[11]. In this section we discuss the behavior of these three protocols for each of the two handover schemes. The following discussion serves as a background for the analysis in Sec. IV and Sec. V.

B. TCP Behavior in the Drop Scheme

The typical behavior of TCP Reno under the *drop* scheme is depicted in Fig. 2. The loss of the dropped packets is discovered by the sender upon the receipt of 3 duplicate ACKs. As a result, the sender enters fast-recovery mode: it reduces *cwnd* by half and retransmits the first dropped packet. The sender exits fast-recovery after receiving an ACK for the retransmitted packet, even if this ACK does not cover all the packets transmitted before entering fast-retransmit. Usually, after 1 or 2 phases of fast-recovery the sender does not receive enough duplicate ACKs to detect the loss of additional packets and it therefore encounters a timeout. In fact, when three or more packets are dropped, a Reno sender almost always encounters a timeout [12]. In Fig. 2 the timeout occurs after a single phase of fast-recovery.

Loss recovery in New-Reno is different. The sender does not exit fast-recovery upon receiving an ACK for the retransmitted copy unless this ACK covers all the packets sent before fast-recovery began. Rather, it sends the next dropped packet and stays in fast-recovery until an impatient timeout [11] The sender is usually able to retransmit 3-4 lost packets in this way before a timeout. After a timeout, the New-Reno sender enters slow-start, exactly like in Reno.

The recovery from handover losses when the connection implements the SACK option is somewhat different. Upon receiving 3 duplicate ACKs – signalling a possible loss – the sender enters fast-recovery and retransmits the missing packets one after the other until all of them reach the mobile node. Then, SACK sender exits fast-recovery phase and continues with congestion avoidance. TCP allows the sender to have no more than $\frac{cwnd}{2}$ outstanding packets during the fast-recovery phase. Therefore, a burst loss of more than $\frac{cwnd}{2}$ packets does not allow a TCP SACK sender to send new packets during the fast-recovery phase, forcing it to wait for a timeout.

C. TCP Behavior in the Forward Scheme

Our discussion of the *forward* scheme begins with New-Reno and SACK. At handover, the packets located along the path between FCP and BS1 are forwarded from BS1 to BS2. The delay of these packets on the forwarding path is likely to cause them to be received by BS2 after some subsequent packets. This is the case shown in Fig. 3. If at least 3 packets arrive at BS2 before the forwarded packets, 3 duplicate ACKs are generated by the receiver, and the sender enters fast-recovery. Consequently, *cwnd* is reduced by half, and the sender retransmits packets that have already been forwarded to the receiver. The sender stays in fast-recovery until it receives an ACK for the last forwarded packet. In the meantime, each duplicate ACK may cause a new packet to be sent if *cwnd* is sufficiently large, and each partial ACK causes an unnecessary retransmission in New-Reno. The difference of SACK is that it retransmits the forwarded packets first, and only then starts sending new packets.

The main difference between Reno and New-Reno/SACK for the *forward* scheme is the shorter fast-recovery phase. After a quick fast-recovery phase, which ends upon reception of an ACK from the first forwarded packet, the *cwnd* in Reno is halved and no new packets are sent until the new window allows it. This prevents many unnecessary retransmissions, presented in New-Reno in Fig. 3.

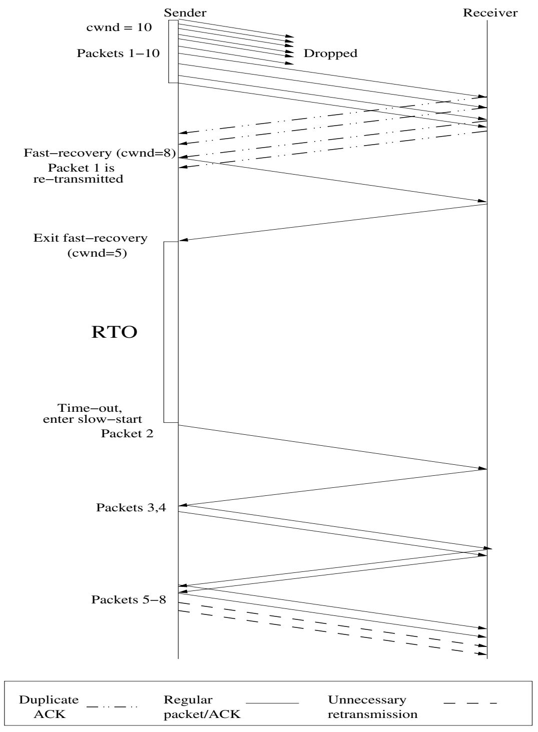


Fig. 2. The *drop* scheme for TCP Reno

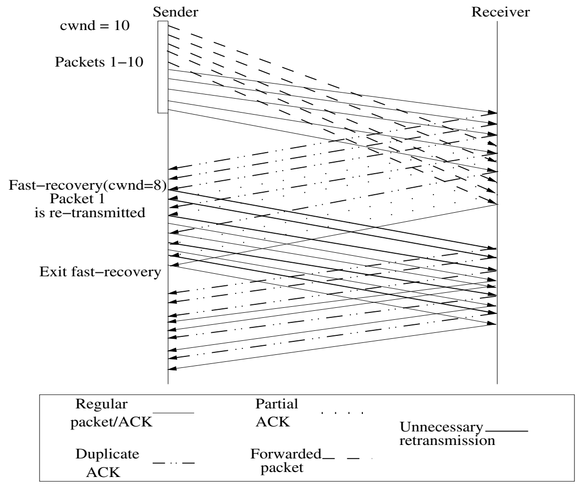
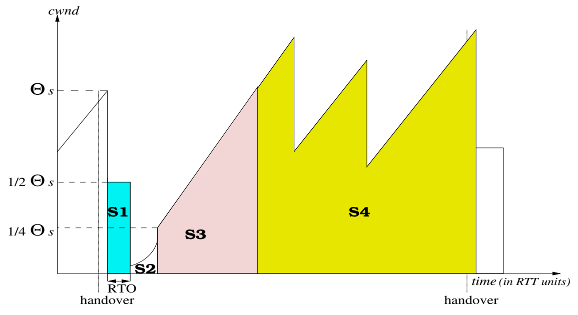


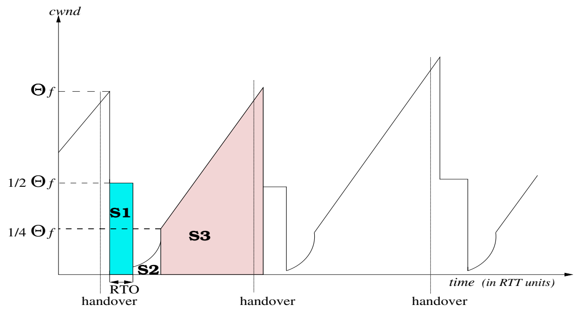
Fig. 3. The *forward* scheme for TCP New-Reno

IV. Analysis of the Drop Scheme

In order to predict the TCP throughput of a mobile host, we distinguish between two cases: the Slow-Mobility (SM) case and the Fast-Mobility (FM) case. In the SM case *cwnd* drops one or more times between handovers (Fig. 4(a)). The SM case is typical when there are many connections sharing bandwidth and infrequent handovers. Hence, the connection is able to reach its TCP fair share, after which it behaves like a regular static connection until the next handover occurs. In



(a) {New-Reno|drop|SM}



(b) {New-Reno|drop|FM}

Fig. 4. The dynamic of *cwnd* for New-Reno under the *drop* scheme

the Fast-Mobility case, *cwnd* does not drop between handovers because it is not able to reach its TCP fair share before the handover takes place (Fig. 4(b)).

We shall use Θ to denote the expected value of *cwnd* when handover takes place. For the FM case, Θ is replaced by Θ_f, while for the SM case it is replaced by Θ_s. Therefore, by [19], $\Theta_s = \frac{3}{4}\overline{CWND}$, where $\overline{CWND} = E[CWND_i]$.

Let the expected total time period between two consecutive handovers be Δ_f, measured in RTT units. The value of Θ_f can be computed if Δ_f is given from the equation:

$$\Delta_f = RTO + (\log_2 \Theta_f - 2) + \frac{3}{4}\Theta_f, \tag{1}$$

. We now estimate the throughput of a New-Reno connection in FM and SM by computing the area bounded by the *cwnd* curve. As it is shown in [19], this area is equal to

$$S_f = \frac{1}{2}\Theta_f(RTO + \frac{1}{2ln2} + \frac{15}{16}\Theta_f) - \frac{1}{ln2}.$$

Unlike in the FM case, in the SM case (Fig. 4(a)) the value of Θ_s is not a function of Δ_s, but rather a property of the specific connection in the specific setting. The computation of the area bordered by the *cwnd* curve gives:

$$S_s = \Theta_s(\Delta_s - \frac{RTO}{2} - log_2\Theta_s + 2 + \frac{1}{4ln2} - \frac{9}{32}\Theta_s) - \frac{1}{ln2}.$$

So far we have approximated the amount of data sent between handovers. However, the data spread between the first common parent (FCP) and the old Base Station BS1 when the handover takes place will be dropped. The amount of this data can be approximated by: $Dropped_{s/f} = D_{\text{FCP}\rightarrow\text{BS}} \times \Theta_{s/f}$, where $D_{\text{FCP}\rightarrow\text{BS}}$ is the delay between FCP and BS1 in RTT units. We can now summarize:

$$Throughput(New\text{-}Reno|drop|FM) = \frac{1}{\Delta_f}\times$$
$$[\Theta_f(\tfrac{RTO}{2} + \tfrac{1}{4ln2} + \tfrac{15}{32}\Theta_f - D_{\text{FCP}\rightarrow\text{BS}}) - \tfrac{1}{ln2}] \tag{2}$$

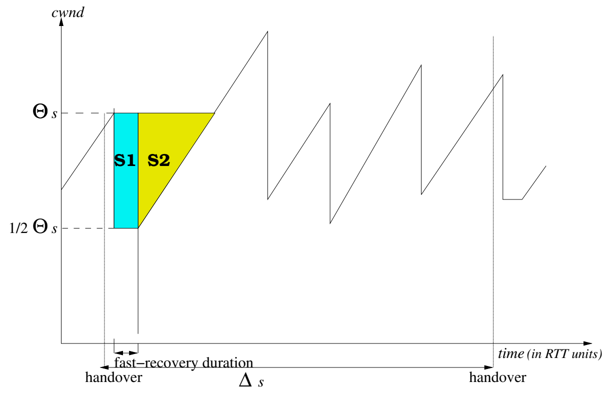


Fig. 5. The dynamic of *cwnd* in {SACK|drop|SM,$D_{\text{FCP}\rightarrow\text{BS}} < \frac{1}{2}$} and in {$*$|forward|SM,fast forwarding}

$$Throughput(New\text{-}Reno|drop|SM) = \tfrac{1}{\Delta_s}\times$$
$$[\Theta_s(\Delta_s - \frac{RTO}{2} - log_2\Theta_s + 2 + \frac{1}{4ln2} - \frac{9}{32}\Theta_s - D_{\text{FCP}\rightarrow\text{BS}}) - \tfrac{1}{ln2}]. \tag{3}$$

Eq. 2 and Eq. 3 can be converted from New-Reno to Reno by incorporating the timeout from additional losses that occur after one or more fast-recovery phases, each lasting one RTT (see Fig. 2). In [12] it is claimed that no more than 3 consecutive fast-recovery phases are likely to take place. In order to simplify the discussion, we assume that there is only one such phase, followed by a timeout, as presented in Fig. 2. Therefore, we get:
$\Delta_f = RTO + 1 + (\log_2 \Theta_f - 2) + \frac{3}{4}\Theta_f$, and

$$Throughput(Reno|drop|FM) = \frac{1}{\Delta_f}\times$$
$$[\Theta_f(\tfrac{RTO+1}{2} + \tfrac{1}{4ln2} + \tfrac{15}{32}\Theta_f - D_{\text{FCP}\rightarrow\text{BS}}) - \tfrac{1}{ln2}] \tag{4}$$

and

$$Throughput(Reno|drop|SM) = \tfrac{1}{\Delta_s}\times[\Theta_s(\Delta_s - \tfrac{RTO}{2} - log_2\Theta_s + \tfrac{3}{2} + \tfrac{1}{4ln2} - \tfrac{9}{32}\Theta_s - D_{\text{FCP}\rightarrow\text{BS}}) - \tfrac{1}{ln2}]. \tag{5}$$

The analysis of SACK is somewhat different. Unlike in Reno and New-Reno, the probability that handover will prevent a SACK sender from entering slow-start is high. A SACK sender enters slow-start only if the number of dropped packets is larger than $\frac{cwnd}{2}$, namely $cwnd \times D_{\text{FCP}\rightarrow\text{BS}} > \frac{cwnd}{2}$, or $D_{\text{FCP}\rightarrow\text{BS}} > \frac{1}{2}$. In this case we have for SACK the same throughput prediction equations as for New-Reno.

When $D_{\text{FCP}\rightarrow\text{BS}} < \frac{1}{2}$, we consider only the Slow-Mobility (SM) case as depicted in Fig. 5. The analysis for FM can be conducted in a similar way. The fast-recovery phase lasts until all dropped packets are retransmitted and acknowledged. Moreover, every partial ACK received during this phase triggers the sending of two additional packets, a process which is very similar to the increase of *cwnd* during slow-start [5]. Therefore, the average time period D_{Ret} during which the sender retransmits the dropped packets can be extracted from the following equation:

$$\int_0^{D_{Ret}} 2^x\,dx = D_{\text{FCP}\rightarrow\text{BS}} \times \Theta_s.$$

Now, we can estimate the connection throughput as

$$Throughput(SACK|drop|SM, D_{\text{FCP}\to\text{BS}} < \tfrac{1}{2}) = \\ = \frac{\Theta_s}{\Delta_s}(\Delta_s - \frac{1}{2}\lceil\frac{1}{\ln 2} \times \ln(D_{\text{FCP}\to\text{BS}} \times \Theta_s \times \ln 2 + 1)\rceil - \\ \tfrac{1}{2} - \tfrac{1}{8}\Theta_s - D_{\text{FCP}\to\text{BS}}). \quad (6)$$

V. Analysis of the Forward Scheme

The main advantage of the *forward* scheme over the *drop* is the shorter time required for the last packet affected by the handover to be acknowledged, as discussed in Sec. III. In fact, if the ACKs are received relatively quickly, it is possible to complete the handover without encountering a timeout and entering slow-start. We therefore distinguish in this analysis between three forwarding speeds: very fast, fast and slow.

With very fast forwarding, the first forwarded packet arrives at the new BS (BS2) before the third packet sent directly to BS2. In the fast forwarding, it is assumed that all the forwarded packets arrive before their retransmissions, as shown in Fig. 3. In the last case, referred to as slow forwarding, it is assumed that the forwarded packets are delayed for a relatively long time and arrive only after their retransmitted copies. The above classification into three cases leaves a grey area between fast and slow forwarding, where some of the first packets arrive after their retransmitted copies, while the last packets arrive during the fast-recovery phase before their retransmissions. In some of these cases the forwarding might still be worthwhile. The merit of forwarding in these cases can be estimated by interpolating the results of fast and slow forwarding.

As already noted, very fast forwarding has almost no effect on the throughput, while slow forwarding has effects similar to those of the *drop* scheme. Therefore, in what follows we analyze only the fast forwarding case. Moreover, we also ignore the combination of Fast-Mobility and *forward*, both because it is rare and because this analysis is very similar to the FM analysis of the *drop* scheme.

Our discussion on fast forwarding for SM begins with New-Reno. The *cwnd* curve for this case is shown in Fig. 5. We assume that the time interval between the receipt of the third duplicate ACK and the acknowledgment of the last forwarded packet is approximately RTO. This time cannot be longer than RTO, of course, but it might be shorter. Again, we predict the throughput for New-Reno by subtracting the penalty due to handover from the throughput without handovers Θ_s. Since $S_1 = \frac{1}{2}\Theta_s \times RTO$ and $S_2 = \frac{1}{2} \times \frac{1}{2}\Theta_s \times \frac{1}{2}\Theta_s = \frac{1}{8}\Theta_s^2$, we have

$$Throughput(\textit{New-Reno}|\textit{forward}|SM, \quad (7) \\ D_{\text{forward}} < D_{\text{FCP}\to\text{BS}} + RTT and \\ \max(D_{\text{forward}}) < RTO) = \tfrac{\Theta_s}{\Delta_s} \times [\Delta_s - \tfrac{RTO}{2} - \tfrac{\Theta_s}{8}].$$

The *cwnd* curve for TCP Reno is similar to New-Reno, as depicted in Fig. 5, except that the fast-recovery phase lasts RTT only. Therefore, we have

$$Throughput(\textit{Reno}|\textit{forward}|SM, \quad (8) \\ D_{\text{forward}} < D_{\text{FCP}\to\text{BS}} + RTT) = \tfrac{\Theta_s}{\Delta_s} \times [\Delta_s - \tfrac{1}{2} - \tfrac{1}{8}\Theta_s].$$

Finally, the analysis of SM for SACK is similar to New-Reno, except that we do not have the limitation of impatient timeout. Therefore,

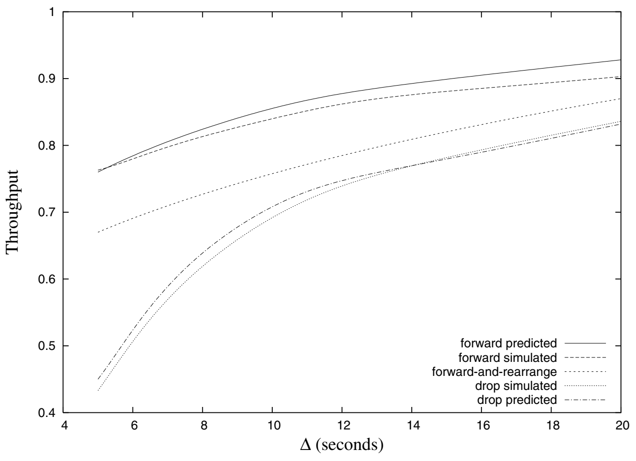


Fig. 6. Throughput for New-Reno as a function of Δ

$$Throughput(\textit{SACK}|\textit{forward}|SM, D_{\text{forward}} < \quad (9) \\ D_{\text{FCP}\to\text{BS}} + RTT) = \tfrac{\Theta_s}{\Delta_s} \times [\Delta_s - \tfrac{1}{2}RTO - \tfrac{1}{8}\Theta_s].$$

VI. Simulation Study

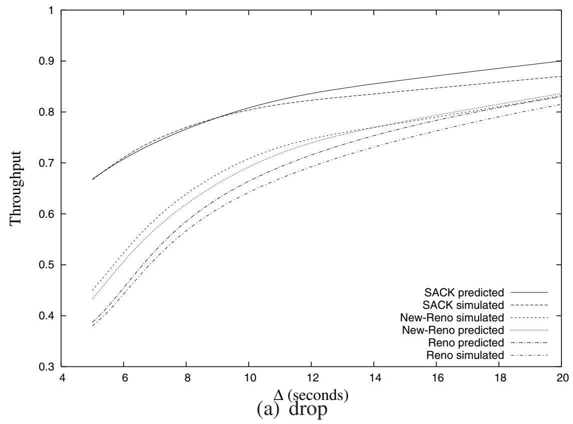
The purpose of the simulation study in this section is twofold: first, to validate the mathematical analysis conducted in Sec. IV and Sec. V, and second, to understand the impact of the various TCP flavors and handover schemes on the performance of a mobile TCP connection.

We developed a simulation model for the network depicted in Fig. 1 using ns-2. We used as a platform the wired (rather than wireless) ns-2 [20] infrastructure, which is renowned for its reliability, and adapted it to our needs by connecting and disconnecting mobile nodes from the network. We start by considering 10 mobile nodes and taking the bandwidth of each wireless link to be 10 times smaller than the bandwidth of the wired links. The propagation delay from the sender to the mobile nodes is 40ms: 25ms between FCP and BS, 5ms between BS and the mobile node, and 10ms on the path between the sender and FCP.

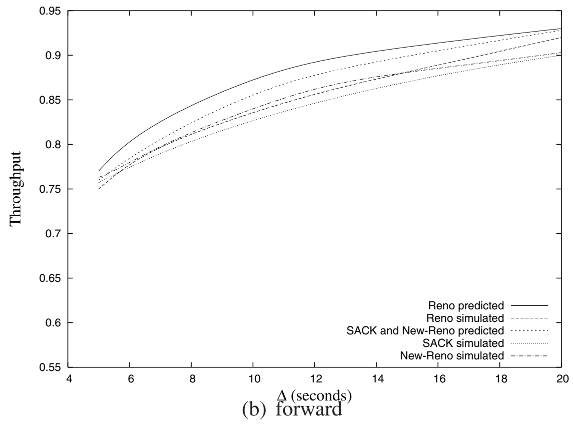
We start with the graph in Fig. 6 This graph depicts the throughput of a mobile host for the New-Reno case as a function of an average handover rate Δ. We consider in the simulations two cases: the case where the bandwidth of each shared backbone link is 100Mb/s while the bandwidth of each private wireless link is 10 Mb/s, and the case where these bandwidths are 1Gb/s and 100 Mb/s respectively (Fig. 6). However, due to the lack of space only the wide bandwidth graph is presented. The graph contains 5 curves: two for the *drop* scheme (predicted vs. simulated), two for the *forward* scheme (predicted vs. simulated), and one for the *forward-and-rearrange* scheme (simulated only) as discussed in Sec. III. Each curve indicates the fraction of bandwidth used for data delivery (i.e., throughput) from the total available bandwidth for each connection. The maximum theoretical throughput 1 indicates 100Mb/s.

The most important conclusions we draw from the graph are as follows:

- The predicted and the simulated results are very close.



(a) drop



(b) forward

Fig. 7. Reno, New-Reno and SACK performance comparison

- As expected, the *forward* scheme outperforms the *drop* scheme by up to 20%.
- Compared to the *forward* scheme, the contribution of the *forward-and-rearrange* scheme might even be negative for the fast backbone! The negative effect is attributed to the fact that the direct packets consume important buffer space at BS2 while waiting for the forwarded packets to be received. Consequently, more packets are lost in BS2. Increasing the buffer space of BS2 slightly decreases the negative impact of *forward-and-rearrange*.

Fig. 7 depicts the performance of New-Reno, Reno and SACK for the 1Gb/s network under the *drop* and *forward* schemes. As before, the precision of our prediction equations is very high. The *forward* scheme for Reno is even better than for New-Reno and SACK (Fig. 7(b)), because there is only one unnecessary retransmission per handover. It is evident that for the *drop* scheme SACK performs significantly better than Reno and New-Reno. As discussed in Sec. IV, this is because the SACK sender is able to retransmit the dropped packets before encountering a timeout.

The additional simulation results, including the influence of the $D_{\text{FCP}\rightarrow\text{BS}}$ on the throughput of a mobile TCP connection and the overhead caused by handover presented in [19].

VII. Conclusions

In this paper we compared the impact of handover on various TCP flavors under two schemes: *drop* and *forward.* In the *drop* scheme, packets received by the base station after the host has disconnected are dropped, whereas in the *forward* scheme these packets are forwarded to the new base station. We developed equations for predicting the loss of throughput encountered by TCP connections due to handover. Our equations take into account several important parameters, such as the handover rate, the relative length of the path from the first common parent of the old and the new base stations to the old base station, and the forwarding speed.

We then conducted simulations using ns-2 in order to validate the mathematical analysis, and in order to understand the impact of the various TCP flavors and handover schemes on the performance of a mobile TCP connection. When we take into account only TCP throughput while ignoring its cost, our main conclusions are that the *forward* scheme is the most attractive one, re-arranging the forwarded packets at the new base-station is not a good idea, and SACK outperforms both Reno and New-Reno. However, when the wireless bandwidth is a major concern, dropping the packets at the old base-station is preferable.

References

[1] C. Bettstetter, H. Vogel, and J. Eberspacher, "GSM phase 2+ general packet radio service GPRS: Architecture, protocols, and air interface," *IEEE Communications Surveys*, vol. 2, no. 3, 1999.

[2] M.Eriksson and et al, "GSM/EDGE radio access network-GERAN; system overview and performance evaluation," *VTC '00*, pp. 2305–2309.

[3] E. Altman, K. Avrachenkov, and C. Barakat, "A stochastic model of TCP/IP with stationary random losses," *IEEE/ACM Transactions On Networking*, vol. 13, no. 2, pp. 356–369, April 2005.

[4] M. Goyal, R. Guerin, and R. Rajan, "Predicting TCP throughput from non-invasive network sampling," *INFOCOM*, 2002.

[5] J. Padhye, V. Firoiu, D. F. Towsley, and J. F. Kurose, "Modeling TCP Reno performance: a simple model and its empirical validation," *IEEE/ACM Transactions on Networking*, vol. 8, no. 2, pp. 133–145, 2000.

[6] A. Gurtov, "Effect of delays on TCP performance," *PWC '01*, pp. 87–108.

[7] J. Schuler and T. Schwabe, "A comparison of the performance of four TCP versions during mobile handoff," *IEEE MWCN'02*, Sept. 2002.

[8] H. Balakrishnan and et al, "A comparison of mechanisms for improving TCP performance over wireless links," *Trans. on Net.*, vol. 5, pp. 756–769, 2000.

[9] V. Tsaoussidis and I. Matta, "Open issues on TCP for mobile computing," *Wireless Communications and Mobile Computing*, vol. 2, pp. 3–20, 2002.

[10] K. Fall and S. Floyd, "Simulation-based comparisons of Tahoe, Reno and SACK TCP," *Computer Communication Review*, July 1996.

[11] S. Floyd and T. Henderson, "The NewReno modification to TCP's fast recovery algorithm," *RFC 2582*, April 1999.

[12] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow, "TCP selective acknowledgement options," *RFC 2018*, October 1996.

[13] A. Kumar, "Comparative performance analysis of versions of TCP in a local network with a lossy link," *Trans. on Net. '98*, vol. 6, pp. 485–498.

[14] M. Mathis, J. Semke, and J. Mahdavi, "The macroscopic behavior of the TCP congestion avoidance algorithm," *Computer Communications Review*, vol. 27, no. 3, 1997.

[15] F. Hu and N. Sharma, "The quantitative analysis of tcp congestion control algorithm in third-generation cellular networks based on fsmc loss model and its performance enhancement," *INFOCOM 2002*, pp. 407–416.

[16] A. Argyriou and V. Madisetti, "Wlc47-1: Modeling the effect of mobile handoffs on tcp and tfrc throughput," *GLOBECOM '06*, pp. 1–5.

[17] C.-T. Chou and K. Shin, "Smooth handoff with enhanced packet buffering-and-forwarding in wireless/mobile networks," *QSHINE '05*.

[18] S. Mohanty and I. F. Akyildiz, "Performance analysis of handoff techniques based on mobile ip, tcp-migrate, and sip," *IEEE Transactions on Mobile Computing*, vol. 6, no. 7, pp. 731–747, 2007.

[19] R. Cohen and A. Levin, "Technical report." [Online]. Available: http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-list.cgi/2007/CS

[20] S.McCanne and S. Floyd, "ns network simulator." [Online]. Available: http://www.isi.edu/nsnam/ns/