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 is the first to present 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: this is, of course, the most important parameter. 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 Section II we present related work. In Section III we describe the various handover schemes in greater detail. In Sections IV and V we analyze the influence of the drop and forward schemes on the TCP throughput of an individual connection. Section VI presents simulation results and compares them to the results computed using our prediction equations. Finally, Section 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. Another experimental work [7] suggests that carefully designed probing mechanisms can cancel incompetent TCP behavior over wireless networks. The authors propose a TCP-Probing modification that prevents a significant portion of the timeouts and unnecessary retransmissions caused by handovers. Ref. [8] 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. A model for optimizing TCP bandwidth use over a noisy wireless link is presented in [9]. The authors propose combining FEC (Forward Error Correction) codes with selective repeat. They present simulation results for different network characteristics that support their throughput prediction formula for Bernoulli distributed errors.

There are also many surveys exploring the problematic aspects of TCP over wireless networks (e.g. [10], [11]). Some of them analyze existing solutions for dealing with TCP throughput degradation due to handovers and lossy links. In [10] 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 [12], New-Reno [13] and SACK [14]. Several works propose models for TCP throughput prediction, e.g., [3], [4], 
[5], [15], [16]. 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 [17] and [18] perform analysis of the drop scheme. In [17], 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 [18], 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 [17], [18] and our work is as follows. In [17], [18], 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 [17], [18], only TCP Reno is analyzed. We also address New-Reno and SACK, and show significant differences in their behavior.

In [19], 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.

In [20], 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 [20], 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 [20].

III. HANDOVER SCHEMES

A. Drop vs. Forward

Figure 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. There are many ways to implement this assumption, e.g., using a GPS-based global time or by giving the control messages high priority, which allows them to be received by the FCPs almost in 0 time. Moreover, our model can also accommodate the case where this assumption does not hold, in the following way. We predict the portion of in-flight packets affected by the handover using the delay from FCP to
BS1, $D_{TCP}$→BS. If FCP modifies its forwarding/routing table some time after the mobile node switches from BS1 to BS2, the portion of affected packets increases. Therefore, by increasing this delay in our prediction equations we can emulate the case where the above assumption does not hold.

The three considered handover schemes – drop, forward, and forward-and-rearrange – are different in their implementation complexity and their effectiveness. In the drop scheme, packets reaching BS1 are silently dropped without any notification to BS2 or the sender. This scheme is the simplest one, but it seems to be inefficient because of the extra bandwidth required for the retransmission of dropped packets and the time spent by the host on timeouts and slow-starts.

In the forward scheme, the packets are forwarded from BS1 to BS2 and then to the mobile node without rearrangement. Compared to drop, this scheme seems to prevent many unnecessary retransmissions, since the forwarded packets eventually arrive at the mobile node. However, some of these packets arrive out-of-order, which may lead to a reduced sending window and to unnecessary packet retransmissions.

The forward-and-rearrange scheme is the most complicated to implement. This scheme requires BS2 to delay new packets received on the direct path from the sender until it sends the host all the packets forwarded by BS1. At first glance this scheme seems to be the most efficient in terms of bandwidth utilization and delivery time minimization. However, our simulation study, presented in Section 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[14], Reno[12] and New-Reno[13]. 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 Section IV and Section V.

B. TCP Behavior in the Drop Scheme

The typical behavior of TCP Reno under the drop scheme is depicted in Figure 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. If the original cwnd was large enough for an additional 3 duplicate ACKs to be received, the sender enters fast-recovery again. 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 [14]. In Figure 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 [13]. 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.

![Fig. 1. Network model](image1)

![Fig. 2. The drop scheme for TCP Reno](image2)
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 Figure 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 Figure 3.

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 (Figure 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\(^1\), after which it behaves like a regular static connection until the next handover occurs. In 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 (Figure 4(b)).

The sender’s reaction to the handover is delayed because the sender discovers packet losses only after receiving three duplicate ACKs. We shall use \( \Theta \) to denote the expected value of \( cwnd \) when the handover takes place. Upon receiving three duplicate ACKs, the TCP sender reduces \( cwnd \) to \( \frac{1}{2} \), and stays with this value until the end of the fast-recovery phase. We assume here that the number of dropped packets is not less than 4, and that a New-Reno sender is not able to overcome all the losses during the impatient timeout period (i.e., the number of losses is larger than impatient timeout/RTT). Therefore, a New-Reno sender enters slow-start after the impatient timeout period. Before entering slow-start, \( ssthresh \) is set to half of the current \( cwnd \), namely to \( \frac{1}{2} \). Then, the value of \( cwnd \) grows exponentially from 1 until it reaches \( ssthresh \). Afterwards, it grows linearly during the congestion-avoidance phase.

Figure 5 depicts the dynamic of \( cwnd \) between two consecutive handovers in the SM case, assuming that timeouts do not happen and that a loss is only indicated by three duplicate ACKs. Let \( T_i \) be a random variable indicating the time period between the \((i-1)\)th burst loss and the \(i\)th burst loss. Denote the value of \( cwnd \) at the end of \( T_i \) by a random variable \( CWND_i \). After reaching this value, \( cwnd \) drops to \( \frac{CWND_i}{2} \), and then grows linearly to \( CWND_{i+1} \) during the next time period \( T_{i+1} \). Hence, the expected value of \( cwnd \) during \( T_i \) is equal to:

\[
\frac{1}{2} (E[CWND_i] + \frac{1}{2} E[CWND_{i-1}]) = \frac{3}{4} E[CWND_i] = \frac{3}{4} CWND,
\]

where \( CWND = E[CWND_i] \).

---

\(^1\)TCP fair share is the bandwidth achieved by the considered TCP connection without handover. We use the term TCP fair share rather than fair share because TCP does not guarantee max-min fairness. Connections that have a larger RTT are able to achieve less than their real fair share, while connections that have a smaller RTT are able to achieve more than their real fair share.
Recall that we 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 Eq. 1, Θ_s = \frac{3}{4} \text{CWND}, where \text{CWND} = E[CWND_i].

We now estimate the throughput of a New-Reno connection in FM and SM by computing the area bounded by the \text{cwnd} curve. Consider the FM case first (Figure 4(b)). Let the expected total time period between two consecutive handovers be Δ_f, measured in RTT units. This time period consists of three parts: the timeout due to the previous handover, the time during which the connection is in slow-start, and the time during which the connection is in congestion-avoidance. As discussed in Section III-B, in New-Reno the first phase lasts RTO. The second phase lasts \log_2 Θ_f RTTs, because \text{cwnd} is doubled every RTT until it reaches the value of \frac{3}{4} Θ_f by the end of the phase. If an ACK is sent for every TCP segment, then, during the congestion-avoidance phase, cwnd will increase by one every RTT while covering a distance of \frac{3}{4} Θ_f. Therefore, the expected total time between two handovers in FM is:

\[ \Delta_f = RTO + (\log_2 Θ_f - 2) + \frac{3}{4} Θ_f, \quad (2) \]

and the value of Θ_f can be computed if Δ_f is given.

In order to predict the connection throughput, we compute the grey area bounded by the \text{cwnd} curve in Figure 4(b). This grey area, denoted S_f, is composed of 3 sub-areas: S_1, S_2 and S_3, where

\[ S_1 = \frac{1}{2} Θ_f \times RTO, \]

\[ S_2 = \int_0^{(\log_2 Θ_f - 2)} \frac{2^x dx}{4 ln 2} = Θ_f - \frac{4}{ln 2}, \]

\[ S_3 = \frac{1}{2} \times \frac{3}{4} Θ_f (\Theta_f + \frac{1}{4}) = \frac{15}{32} Θ_f^2. \]

Therefore,

\[ S_f = \frac{1}{2} Θ_f (RTO + \frac{1}{2 ln 2} + \frac{15}{16} Θ_f) - \frac{1}{ln 2}. \]

Unlike in the FM case, in the SM case (Figure 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 \text{cwnd} curve is similar to that conducted earlier for the FM case with Θ_s and Δ_s replacing Θ_f and Δ_f respectively, except that we have an additional component S_1 (see Figure 4(a)). We approximate this area by a rectangle whose height and width are Θ_s and Δ_s−T_f respectively, where T_f is equal to Δ_f with Θ_s replacing Θ_f. Therefore, by Eq. 2 we get

\[ S_s = Θ_s (Δ_s - RTO - (\log_2 Θ_s - 2) - \frac{3}{4} Θ_s). \]

Simple geometrical considerations reveal that the precision of this approximation is perfect when \text{CWND}_i has the same value for every i, which is of course unlikely. However, it turns out that the precision is excellent unless the variance of \text{CWND}_i is very high, which is unlikely as well.

If we use the FM equations for S_1, S_2 and S_3 while substituting Θ_f with Θ_s, we get:

\[ S_s = Θ_s (Δ_s - RTO - (\log_2 Θ_s - 2) - \frac{3}{4} Θ_s) - \frac{1}{ln^2}. \]

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_{FCP→BS} × Θ_{s/f}, where D_{FCP→BS} is the delay between FCP and BS1 in RTT units. We can now summarize:

\[ Throughput(\text{New-Reno}|\text{drop}|\text{FM}) = \frac{1}{Δ_f} × \left[ Θ_f \left( \frac{RTO}{2} + \frac{1}{4 ln 2} + \frac{15}{32} Θ_f - D_{FCP→BS} - \frac{1}{ln 2} \right) \right] \]

\[ Throughput(\text{New-Reno}|\text{drop}|\text{SM}) = \frac{1}{Δ_s} × \left[ Θ_s (Δ_s - RTO - (\log_2 Θ_s - 2) - \frac{3}{4} Θ_s) - D_{FCP→BS} - \frac{1}{ln 2} \right]. \]

where Θ_f is computed from Eq. 2 and Θ_s = \frac{3}{4} \text{CWND}. We assume that, when the handover takes place, the mobile node is able to receive the packets that have already been transmitted by BS1. Without this assumption, D_{FCP→BS} should be replaced by D_{FCP→mobile}.

Eq. 2, Eq. 3 and Eq. 4 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 Figure 2). In [14] 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 Figure 2. This implies that the width of S_1 is RTO+1, instead of RTO in New-Reno, and the width of S_4 is reduced by 1 to Δ_s - RTO - 1 - (\log_2 Θ_s - 2) - \frac{3}{4} Θ_s, where time is measured again in RTT units. Therefore, we get:

\[ Δ_f = RTO + 1 + (\log_2 Θ_f - 2) + \frac{3}{4} Θ_f, \]

\[ Throughput(\text{Reno}|\text{drop}|\text{FM}) = \frac{1}{Δ_f} × \left[ Θ_f \left( \frac{RTO+1}{2} + \frac{1}{4 ln 2} + \frac{15}{32} Θ_f - D_{FCP→BS} - \frac{1}{ln 2} \right) \right] \]

and

\[ Throughput(\text{Reno}|\text{drop}|\text{SM}) = \frac{1}{Δ_s} \times \left[ Θ_s (Δ_s - RTO - (\log_2 Θ_s - 2) - \frac{3}{4} Θ_s) - D_{FCP→BS} - \frac{1}{ln 2} \right]. \]
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. As discussed in Section III-B, a SACK sender enters slow-start only if the number of dropped packets is larger than $cwnd$.

The condition for entering slow-start, namely $cwnd \times D_{\text{FCP} \rightarrow \text{BS}} > cwnd$, is translated to $D_{\text{FCP} \rightarrow \text{BS}} > \frac{1}{2}$. 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 Figure 6. 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 $D_{\text{Ret}}$ during which the sender retransmits the dropped packets can be extracted from the following equation:

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

This implies that

$$D_{\text{Ret}} = \left\lceil \frac{\ln(D_{\text{FCP} \rightarrow \text{BS}} \times \Theta_s \times \ln 2 + 1)}{\ln 2} \right\rceil.  \tag{7}$$

The duration of fast-recovery is equal to $D_{\text{Ret}} + 1$, because one RTT is needed in order to receive an ACK for the last retransmitted packet. Now, we can estimate the connection throughput by computing the area under the $cwnd$ curve in Figure 6 and then subtracting the amount of retransmitted data. The area under $cwnd$ curve can be estimated by the area under the horizontal line $\Theta_s$, i.e., $\Theta_s \times \Delta_s$, minus the grey areas $S_1$ and $S_2$ in Figure 6. Since $S_1 = \frac{1}{2} \Theta_s \times (D_{\text{Ret}} + 1)$ and $S_2 = \frac{1}{2} \times \frac{1}{8} \Theta_s = \frac{1}{8} \Theta_s^2$, we get

$$\text{Throughput}(SACK|\text{drop}|SM, D_{\text{FCP} \rightarrow \text{BS}} < \frac{1}{2}) = \frac{\Theta_s \times (\Delta_s - \frac{1}{2} \ln 2)}{\Delta_s} \times \Theta_s \times \ln 2 + 1) - \frac{1}{2} - \frac{1}{8} \Theta_s - D_{\text{FCP} \rightarrow \text{BS}}. \tag{8}$$

V. ANALYSIS OF THE FORWARD SCHEME

The main advantage of the forward scheme over the drop scheme is the shorter time required for the last packet affected by the handover to be acknowledged, as discussed in Section 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. Therefore, 3 duplicate ACKs are not received in this situation, and the minor reordering of the packets has almost no impact on the TCP throughput. We assume that when the handover takes place the first forwarded packet is located near BS1, while the first packet not affected by the handover, referred to as the first direct packet, is located just before the FCP. Therefore, the condition for very fast forwarding is: $D_{\text{forward}} < D_{\text{FCP} \rightarrow \text{BS}}$, where $D_{\text{forward}}$ is the delay of the forwarded packet on the forwarding path between BS1 and BS2. This condition ignores the transmission time of two additional direct packets. We will not further analyze this case because of the negligible impact the handover has here on the throughput.

In the next case, referred to as fast forwarding, it is assumed that all the forwarded packets arrive before their retransmissions, as shown in Figure 3. As a result, 3 duplicate ACKs are received by the sender, and $cwnd$ is reduced by half. However, there is no timeout since the forwarded packets arrive one after the other even before their retransmitted copies. Therefore, the condition for fast forwarding, for all TCP variants we discuss, is: $D_{\text{forward}} < D_{\text{FCP} \rightarrow \text{BS}} + RTT$.

However, for New-Reno there is an additional condition caused by its impatient timeout, which expires $D_{\text{FCP} \rightarrow \text{mobile}} + D_{\text{ACK}} + RTO$ after handover, where $D_{\text{ACK}}$ is the end-to-end delay of the ACK messages. The ACK for the last forwarded packet is received by the sender at $\max(D_{\text{forward}}) + D_{\text{FCP} \rightarrow \text{mobile}} + D_{\text{ACK}}$ after the handover takes place, where $\max(D_{\text{forward}})$ is the time it takes for the last forwarded packet to traverse the forwarding path from BS1 to BS2. Hence, the additional condition for fast forwarding in New-Reno is: $\max(D_{\text{forward}}) + D_{\text{FCP} \rightarrow \text{mobile}} + D_{\text{ACK}} < D_{\text{FCP} \rightarrow \text{mobile}} + D_{\text{ACK}} + RTO$, or simply $\max(D_{\text{forward}}) < RTO$.

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. From the sender’s perspective, this case is very similar to drop. Throughput prediction for slow forwarding is similar to the drop scheme, except that the retransmission time of the forwarded packets might be shorter.

Slow forwarding in TCP Reno proceeds as follows. The TCP sender enters fast-recovery upon receiving 3 duplicate ACKs due to out-of-order packet arrival. Fast-recovery ends when an ACK for the retransmitted packet is received, as shown in Figure 2. Just like in the drop analysis, we consider only the case of a single fast-recovery phase, in order to avoid getting into exhausting details whose effect on the throughput is negligible. If the forwarded packets are not acknowledged by the end of the fast-recovery phase, then the forwarding is said to be slow, and the connection enters slow-start. Hence,
the condition for slow forwarding in Reno is $D_{\text{forward}} > RTO + RTT$.

In New-Reno, slow forwarding means that the forwarded packets are not able to reach the mobile node and be acknowledged by the end of the impatient timeout. Therefore, the condition for slow forwarding in this case is $\max(D_{\text{forward}}) > RTO$.

Unlike in Reno and New-Reno, slow forwarding in SACK does not necessarily trigger a timeout. In order for the forwarded packets to arrive after their retransmissions, the following condition should be fulfilled:

$$D_{\text{forward}} > \left| \ln(D_{\text{FCP-BS}} \times \Theta_s \times \ln 2 + 1) \right|.$$  

The right side of this equation represents the time it takes for SACK to retransmit the forwarded packets, as computed in Eq. 7 for the drop scheme.

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 Figure 6. 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 $\Theta_s$. 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

$$\text{Throughput}(\text{New-Reno}[\text{forward}]\text{SM}),$$

$$D_{\text{forward}} < D_{\text{FCP-BS}} + RTT \quad \text{and} \quad \max(D_{\text{forward}}) < RTO \Rightarrow \frac{\Theta_s}{\Delta_s} \times [\Delta_s - \frac{RTO}{2} - \frac{\Theta_s}{8}].$$

Recall that the condition for fast forwarding in TCP Reno is: $D_{\text{forward}} < D_{\text{FCP-BS}} + RTT$. The cwnd curve for this case is similar to New-Reno, as depicted in Figure 6, except that the fast-recovery phase lasts $RTT$ only. Therefore, we have

$$\text{Throughput}(\text{Reno}[\text{forward}]\text{SM}),$$

$$D_{\text{forward}} < D_{\text{FCP-BS}} + RTT = \frac{\Theta_s}{\Delta_s} \times [\Delta_s - \frac{1}{2} - \frac{1}{8} \Theta_s].$$

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

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

VI. SIMULATION STUDY

The purpose of the simulation study in this section is twofold: first, to validate the mathematical analysis conducted in Sections IV and 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 Figure 1 using ns-2. We used as a platform the wired (rather than wireless) ns-2 [21] 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. When handover rate is high (FM), bandwidth utilization is small and there is no queueing delay in the network. In that case RTT is equal to the propagation delay. However, when the mobility rate is not high (SM), each connection increases its bandwidth until congestion is encountered. Consequently, RTT is mostly affected by the queueing delay in the various links and we measure the simulated RTT. The value of RTO is approximated in our equations by $3^*RTT$. This approximation is reasonable because in our model handover is very quick and therefore there are no consecutive timeouts.

As the time interval between two consecutive handovers in the simulations is uniformly distributed. In our equations, $\Delta$ is taken to be the expected value of this distribution. As in the performance analysis, we assume here as well that the FCP is notified in zero time about the handover. The value of $\Theta_s$ for the SM case, which actually indicates the connection throughput without handovers, is approximated for the sake of our prediction as the max-min fair share of the connection. This approximation is valid because in our model all the connections have the same end-to-end delay and they all encounter congestion in the same link – the one originating at the sender. In order to determine whether the SM or the FM prediction equation is applicable for each specific setting, we compute the time between handovers and check if it allows the connection to reach its fair share. If it does, it is considered to be FM. Otherwise, it is considered to be SM. It turns out that most of the points on each graph are attributed to SM, and only 3-4 of them are attributed to FM. Throughout this section, when we address the forward scheme, we refer to the fast forwarding case only.

We start with the graph in Figure 7. This graph depicts the throughput of a mobile host for the New-Reno case as a function of an average handover rate $\Delta$. 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 (Figure 7). 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 Section 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 curves of the predicted bandwidth are computed using the equations for the SM case, except when \( \Delta < 7 \), where the condition of FM is applicable. The most important conclusions we draw from the graph are as follows:

1) The predicted results are very close to the simulated results.
2) As expected, the forward scheme outperforms the drop scheme by up to 20%.
3) 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. When the buffer space of BS2 increases, we see that the negative impact of forward-and-rearrange decreases slightly.

Figure 8 depicts the performance of New-Reno, Reno and SACK for the 1Gb/s network under the drop and forward schemes. As before, it is evident that the precision of our prediction equations is very high. Recall that the forward prediction equations for SACK and New-Reno are the same. Hence, they are represented by a single curve. The forward scheme for Reno is even better than for New-Reno and SACK (Figure 8(b)), because there is only one unnecessary retransmission per handover. Note, however, that the results of Reno are very sensitive to the size of the base-station buffer. The simulation results shown in Figure 8(b) reflect the case where the size of this buffer is 250KB per mobile node. If this size is reduced, e.g., to 50KB, the simulation results for Reno are significantly worse than the results of SACK and New-Reno. This is explained by the sensitivity of Reno to packet losses during handover, losses which may be caused by temporary congestion in the buffer of BS2 due to the forwarded packets, where each packet loss leads to an additional halving of cwnd.

In New-Reno and SACK this additional cwnd reduction is less likely, because they are very likely to cover the lost packet in their additional retransmissions during the fast-recovery phase.

It is evident that for the drop scheme SACK performs significantly better than Reno and New-Reno. As discussed in Section IV, this is because the SACK sender is able to retransmit the dropped packets before encountering a timeout. The performance of New-Reno is slightly better than Reno because it takes longer for a timeout to be triggered. As discussed in Section IV, it takes \( \text{RTT} + \text{RTO} \) in Reno, but only RTO in New-Reno. Also, as discussed in Sections IV and V, it is evident from the graph that even though the SACK sender does not enter slow-start in drop or in forward, its performance under forward is still better. The reason for this is the longer fast-recovery phase, as described in Figure 6.

Another major influence on the throughput of a mobile TCP connection is the delay \( D_{\text{TCP-BS}} \). Recall that by increasing this delay we can also simulate the case where the time it takes to inform the FCP about the handover is not 0. Figure 9 shows the simulated throughput for Reno, New-Reno and SACK as a function of this delay for 10 mobile hosts and the 1Gb/s backbone network. As mentioned in our analysis, this delay defines the fraction of in-flight packets lost in the drop scheme or forwarded in the forward scheme. This explains
the reduction of the *drop* curves for all TCP flavors. The particularly steep fall of the SACK curve is explained by the fact that the probability of the dropped data to reach $\frac{1}{2} \times cwnd$ increases. As mentioned in Section IV, this is the maximum ratio for avoiding SACK timeouts.

It is interesting to note that in contrast to the *drop* curves, which decrease monotonically with the delay $D_{\text{FCP->BS}}$, the *forward* curves demonstrate a slight drop followed by a significant improvement. The slight drop is attributed to the fact that more packets are affected by the handover, and the likelihood for out-of-order delivery increases. However, from some point in *forward*, there will be very few direct packets because, when the handover takes place, most of the in-flight packets are located between FCP and BS1 rather than between the sender and FCP. Consequently, the forwarded packets do not trigger enough duplicate ACKs, thereby simulating the case of very fast forwarding, which yields a better throughput.

In the discussion so far we have only addressed the throughput of an individual host or that of all the hosts together. We have not addressed another important factor, namely the total bandwidth consumed by the network in order to achieve that throughput. This bandwidth is affected by the number of dropped packets (i.e., the number of necessary retransmissions), the number of unnecessary retransmissions, and the bandwidth cost of forwarded packets. Figure 10 shows the percentage of overhead caused by handover as a function of RTT in TCP SACK and New-Reno for the *drop* and *forward* schemes.

As mentioned in our analysis, the RTT, and especially the delay between the FCP and the base stations, affect the amount of in-flight packets that are lost in the *drop* scheme or forwarded in the *forward* scheme. Figures 2 and 3 illustrated the cases where some of these packets are unnecessary retransmitted, consuming extra wireless bandwidth. The wireless channels are usually more expensive, vulnerable and congested than the wired network, so they require a special attention. Figure 10 shows that the overhead grows with the RTT and that the *forward* scheme consumes more extra bandwidth than the *drop* scheme. Also, it is obvious that the selective acknowledgment option saves most of the unnecessary retransmissions both for *drop* and *forward*. The main conclusion from Figure 10 is that when TCP SACK is used, the wireless penalty for *drop* and *forward* is almost the same, and therefore should not be a factor. In contrast, when New-Reno is used, the wireless penalty of the *forward* scheme is a concern when the RTT is relatively long.

The most general observations we can make from our study as far as the throughput is concerned are as follows:

- The rather light complexity required for the implementation of the *forward* scheme is worthwhile.
- The rather heavy complexity required for the implementation of the *forward-and-rearrange* scheme is not worthwhile at all.
- Out of the three considered TCP variants, SACK is the best option. Implementing SACK is more important with the *drop* scheme and less important with the *forward* scheme. The small advantage of Reno over SACK for the *forward* scheme is somewhat misleading because Reno’s throughput is much more sensitive to the size of the base station buffer.

### 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


