Fast Link Adaptation for TFRC after a Handover

Dagang Li, Kristof Sleurs, Emmanuel Van Lil and Antoine Van de Capelle
TELEMIC, Dept. of Electrical Engineering-ESAT, Katholieke Universiteit Leuven
Kasteelpark Arenberg 10, B-3001 Heverlee, Belgium
Email: firstname.lastname@esat.kuleuven.be

Abstract—TCP-Friendly Rate Control(TFRC) is an equation-based congestion control mechanism that competes fairly with TCP but has a much lower throughput variation, which makes it a better choice for streaming over the Internet. It is known that in a mobile network environment, after a handover TFRC can overshoot or underutilize the new link if the conditions there differ from those of the old link. There are different factors that affect the TFRC performance during a handover. In this paper we focus on the impact of the change in the round trip time, an aspect that is largely overlooked in the literature compared with bandwidth disparity. We find that even if the same bandwidth is assured on the new link, changes in the link latency will still cause TFRC performance degradation after the handover. A fast link adaptation mechanism is thus proposed to address the problem. Simulations show that this mechanism helps TFRC to adjust quickly to the new link without consecutive packet loss or long-time bandwidth underutilization.

Index Terms—TFRC, Handover, Mobility, RTT, Latency.

I. INTRODUCTION

The lack of congestion control in most of today's real-time and multimedia flows means that they are unresponsive to the changes in the network status and do not compete fairly with congestion-controlled traffic, mainly run on TCP, in the best-effort Internet. Uncooperative traffic is harmful to the Internet, especially when its volume is growing by time. Unfortunately, TCP is not suitable for these flows since it is a connection-oriented protocol and uses retransmission for reliable packet delivery. These features have penalty on timeliness therefore are undesirable for real-time and multimedia applications. One way to make these flows responsive to congestion is to apply only the congestion control *mechanism* used by TCP, namely Additive-Increase Multiplicative-Decrease (AIMD) scheme, without using the *protocol* itself. Flows controlled by AIMD will naturally compete fairly with TCP, but they also preserve the typical TCP behavior of halving the sending rate in response to a single packet drop. This behavior can be too severe for applications like telephony and streaming that require a smooth sending rate.

TCP-Friendly Rate Control (TFRC) [1] is designed to compete fairly with TCP, but at the same time has a much lower variation in throughput compared with TCP and other AIMD based mechanisms. The fairness comes from the fact that TFRC uses the TCP throughput equation to calculate the allowed sending rate. The smoothness lies in the way how the variables of this equation are updated: basically past history is used to weight out the instantaneous variation in the latest samples. The cost of a smoother throughput is slower

response to changes that are not short-term variations: for the abrupt network change during a handover, the long response time might lead to overshoot or underutilization problems. Although handover causes problems for TCP as well, solutions for TCP can not be applied to TFRC directly, because TCP and TFRC use different congestion control principles.

The most apparent handover-induced problem is caused by a different available bandwidth in the new network. The effect of a changed round-trip time (RTT), which is equally significant as shown in this paper, is nevertheless largely overlooked. Indeed, the sending rate of TFRC is also a function of RTT that may suffer abrupt change during a handover as well, due to the changes in routing, link latency, competing flows, etc. Furthermore, since RTT is interrelated with other network factors, the impact of an abrupt RTT change is quite dynamic and deserves careful study, thus the goal of this paper.

In order to focus on RTT changes, in this paper we assume the same *available* bandwidth before and after a handover. This assumption is typical for horizontal handovers, but it is also possible during a vertical one, due to QoS methods such as reservation, prioritization, resource allocation, or even due to the competition from cross traffic. The study on bandwidth change is presented in [7]. The rest of the paper is organized as follows. We start with the background information and an overview of the related work in Section 2. In Section 3 the behavior of TFRC during a RTT-changed-only handover is analyzed. Based on the observations from this section, a fast link adaptation mechanism is proposed in Section 4 to improve the TFRC performance. The proposed technique is evaluated in Section 5 with simulations. Section 6 concludes the paper.

II. BACKGROUND AND RELATED WORK

TFRC [2] [1] is an equation-based congestion control protocol that maintains a relatively smooth sending rate which is generally within a factor of two of the sending rate of a TCP flow under the same conditions.

A. Equation-based Congestion Control

The recommended equation used in TFRC is a slightly simplified version of the throughput equation for Reno TCP from [3], as shown below in Eq.1. It describes the sending rate X as a function of packet size s , RTT R , TCP retransmission timeout rto , loss event rate p and finally b , the number of packets acknowledged by one ACK. A loss event is defined

as the aggregation of all the packet losses within one RTT.

$$X = \frac{s}{R\sqrt{\frac{2bp}{3}} + rto \cdot 3\sqrt{\frac{3bp}{8}}p(1 + 32p^2)} \quad (1)$$

In most cases, the equation can be simplified by letting $rto = 4R$ and $b = 1$ [1]. Since s is normally known to an application, within each TFRC flow, Eq.1 can be rewritten as

$$X = s \cdot f(p)/R \quad (2)$$

with only two variables R and p , where $f(p) = (\sqrt{2p/3} + 12\sqrt{3p/8} \cdot p(1 + 32p^2))^{-1}$. This means that generally, a TFRC flow can only sense the network conditions through RTT and the loss event rate. If these two differ greatly from the reality, TFRC will undoubtedly fail to behave appropriately.

B. Rate Control in TFRC

The primary goal of TFRC is to maintain a relatively steady sending rate while still being responsive to congestion, therefore it is designed not to aggressively seek out available bandwidth. For the same reason, TFRC also does not halve the sending rate in response to a single packet loss, but it should still back off quickly in case of successive loss events. These goals are nicely achieved by using Exponentially Weighted Moving Average (EWMA) to update the RTT R and Weighted Average Loss Interval (WALI) to calculate loss event rate p . EWMA smooths out the short-term turbulence in the RTT samples and only follows the long-term trend; on the other hand, WALI keeps a small number of recent loss intervals and gives higher weight to the latest ones when calculating the average. The loss event rate p is then the reciprocal of the average loss interval.

Moreover, in order to be fair with TCP, further limits are applied to the calculated rate X from Eq.1. First, if no feedback is received for rto , it will be cut in half; second, it cannot increase faster than 1 packet per RTT. Because there is no self-clocking mechanism embedded in TFRC, the allowed sending rate is also explicitly limited by twice the reported receiving rate. At last, oscillation reduction from Eq.3 is applied and the final result X_{inst} is used to schedule packets.

$$X_{inst} = X \cdot (R_{sqmean}/\sqrt{R}) \quad (3)$$

where R_{sqmean} is the EWMA of the square root of the RTT samples.

C. TFRC Handover

Just like TCP, TFRC is not designed with mobility in mind. This is quite obvious from the way how RTT and loss event rate are updated. In a mobile environment, a handover will very likely cause abrupt change in the network conditions, but EWMA is known to be slow when responding to such a step change. On the other hand, WALI only maintains a limited record set that can be fully updated with just a few new samples. During a packet loss increase, enough loss interval samples can be easily obtained to drive the WALI algorithm; however, when packet loss decreases greatly, it will take a long time before a new packet drop emerges to conclude the

current loss interval and update the record in WALI. Before enough updates are collected, WALI will keep staying out of track.

The problem with TFRC handover has already been recognized and studied in the literature, but mostly around bandwidth disparity. In [4] the performance of TFRC during vertical handovers is measured in a testbed and then analyzed with simulations. The authors suggest to reset the state of the TFRC receiver after a handover and send adjusted feedback for 3 RTTs to inform the sender: when the new link is faster, reporting error losses is suppressed and the receive rate is set according to the new link's rate; when the new link is slower, the average loss rate is artificially increased to force the sender to reduce the rate faster. The algorithm to calculate the adjustment is however not discussed in the paper. In [6] a detailed mechanism is presented using packet-pair to detect the bandwidth change caused by a handover. When a bandwidth increase is detected, they just let TFRC go back to *slow start* to probe for the new available bandwidth; when the bandwidth decreases, an urgent message is sent to the sender with the estimated bandwidth from the detection mechanism to limit the sending rate. The side effect of going back through slow start is the large sending rate variance and packet loss during bandwidth probing, as we will see later. If both the old and new links are accessible during a handover, authors of [5] propose to probe the available bandwidth of the new link before a handover is performed. However, the technique needs the support to multi-path transmission and major changes in the TFRC protocol, which bring difficulties to wide deployment.

After all, none of the previous work has looked into the effect of a changed RTT. Without specific concern on possible RTT disparity in their design, these techniques are less efficient when the handover also involves RTT changes.

III. TFRC HANDOVER WITH RTT IMPARITY

In this section we will study the effect of a changed RTT on the TFRC performance during a handover. As mentioned earlier, we assume the same available bandwidth across the handover. RTT can either increase or decrease, both shown in Fig.1. In this figure, the available bandwidth is always 5Mbps, and RTT changes between about 50ms and about 150ms. The bigger RTT variation along the higher delayed link is due to the bigger queueing dynamics because of the larger bandwidth-delay product (BDP). The network topology is the same as in Fig.4 with the same settings from Section 5.

Fig.1(a) and 1(c) show the evolution of TFRC calculated sending rate X , real sending rate X_{inst} and the receiving rate X_{rcv} , with packet loss marked on the time line. RTT and the loss event rate p are instead shown in Fig.1(b) and 1(d).

Because the bandwidth limit is not changed, from Eq.1 we would expect the loss event rate p to change in the opposite direction of RTT, which is clearly seen in Fig.1(b) and 1(d). On the other hand, Eq.2 also tells us that the TFRC behavior in this transition period is determined by the speed and accuracy of the adaptation with RTT and the loss event rate p

Fig. 1. TFRC Performance during a Handover with changed RTT

A. Case 1: Low RTT to High RTT

Fig.1(a) shows that after the handover there is a long period of underutilization. This is caused by a sudden drop in the sending rate right after the handover followed by a slow growth until the bandwidth limit is reached again. The sudden drop is caused by the increase of RTT, since at that moment p has not changed and X is a strict decreasing function of R as shown in Eq.2. The real sending rate X_{inst} decreases even further because of the oscillation reduction from Eq.3. Indeed, the new RTT sample is far higher than R_{sqmean} , which drags the X_{inst} further down.

The following slow rate increase is more as a result of the slow p decrease shown in Fig.1(b), since X is also a strict decreasing function of p . This is typical for WALI, because when the link is underutilized, packet loss is much less likely to happen.

B. Case 2: High RTT to Low RTT

Fig.1(c) shows that while the throughput recovers quickly, the main problem now is the excessive packet loss during the handover. When RTT decreases, the sending rate already at full speed will immediately go beyond the bandwidth limit and cause packet loss. With a lot of packet loss, WALI can

easily collect enough loss interval samples to quickly adapt p , as shown in Fig.1(d). But since all the packet losses within one RTT are counted as just a single loss event, with an outdated RTT estimation that is clearly too big, more packets are lost during the period, numbered to 114 in this simulation. Here the oscillation reduction again amplifies the negative effect of a lagged RTT adaptation: it drives the X_{inst} so high that X has to go deep under the available bandwidth to compensate, as shown in Fig.1(c).

It is also shown in Fig.1 that although *slow start* can probe the available bandwidth in a short time, it causes high throughput variation and burst of packet loss as well. During a handover, a smooth throughput is very often more important than to aggressively utilize the new bandwidth, therefore the method from [6] is not suitable for these cases. On the other hand, since RTT is maintained at the TFRC sender, techniques that simply resets the receiver as in [4] will still suffer from the slow RTT adaptation.

IV. FAST LINK ADAPTATION MECHANISM

From the study above we have the following observations:

- 1) During a handover, the change in RTT always drive the sending rate to the opposite direction, regardless of the real bandwidth in the new link. The effect is again amplified by the oscillation reduction procedure which makes inaccurate RTT estimation even more harmful.
- 2) The loss event rate p increases fast but decreases slowly. Because of this an under-rated TFRC flow will take a long time to reach the full throughput.
- 3) Starting over and reprobng after each handover is fast to adapt to the new link, but it also brings throughput variation and excessive packet loss.

Based on these observations, we decide to abandon the unreliable R and p estimation all together during a handover and keep the sending rate unchanged, until accurate samples can be collected from the new link again. If by some means not discussed in this paper (through handover prediction or periodically status info exchange, for example), the available bandwidth BW on the new link is known in advance, then the sending rate can be kept at BW instead. When reliable samples become available, R can be directly set to the sample value and resume EWMA updating from there. Since samples for p is not easy to collect, it will resume WALI updating from an artificial state constructed around the desired sending rate (be it the old steady rate or BW) and the adjusted RTT. After this time the fast link adaptation is finished and TFRC can behave in the normal way again.

Our mechanism is designed to specifically handle changed RTT during a handover. It leaves the problem of bandwidth disparity to other (available) techniques and can work together with them if necessary. Another feature of the mechanism is that it does not differentiate RTT increase and decrease. The same procedure is followed in both cases, as shown in Fig.2. The details are discussed below.

Fig. 2. Fast Link Adaption Mechanism

A. Message Exchange

Our design follows the TFRC convention not to burden the sender. All critical decisions are made at the receiver. We assume a 3rd-layer lossless handover technique is used, so that packets still destined to the old link after a handover will not get lost. Examples of such techniques are *buffer-and-forward* and *make-before-break*. These packets can be easily identified from other data packets at the receiver by means of tunnel marking or interface distinction.

As shown in Fig.2, four time points separate the handover into several phases. In each phase the receiver manipulates the feedback packets differently to interact with the sender.

- t_0 The first packet from the old link is received after a handover. This packet serves as the indication of the handover. From now on the receiver sends an urgent feedback with handover notification (HOn) to the sender upon each data packet received, with p set to 0 and X_{rcv} to half of the old sending rate. They are sent along the new link, so the RTT measured on them would be about $RTT_o/2 + RTT_n/2$. They reset the RTT record at the sender and keep the sending rate steady at the old rate. These packets are marked in blue in the figure.
- t_1 The first packet from the new link is received, and the receiver sends out one last urgent update to reset the RTT record, with the first correct RTT_n sample. After it no more urgent feedback is generated. For the following periodical updates, p is still set to 0 and X_{rcv} to the old rate as before to maintain the steady sending rate.
- t_2 As we will see soon, upon receiving an urgent feedback with HOn, the sender marks all the following new data packets as *echo* (shown in orange in Fig.2). t_3 is the time first such packet reaches the receiver. All the echo packets carry an RTT estimation of $RTT_o/2 + RTT_n/2$. With RTT_o known from the past, the receiver can now calculate RTT_n . Base on this estimated RTT_n and the desired sending rate, an artificial p can be calculated from Eq.1 and used to reconstruct the history set of loss intervals at the receiver. From now on the real receiving rate X_{rcv} and the artificial p are used in the feedback.
- t_3 The first non-echo (aka normal) packet is received. This indicates the end of fast link adaptation and TFRC resumes the normal behavior from now on. Indeed, after the previous phases TFRC has settled into a state defined by RTT_n and the desired sending rate, which serves as

Fig. 3. TFRC Receiver State Transition Graph

a good starting point for TFRC in the new link.

B. Receiver Algorithm

The algorithm at the receiver is best presented by the state transition diagram shown in Fig.3. State 0–3 corresponds to the period of t_3-t_0 , t_0-t_1 , t_1-t_2 and t_2-t_3 respectively. The behavior of the receiver in each state has already been briefly discussed in the previous subsection with message exchange. It is summarized in Fig.3 with the state transition conditions and actions.

One thing we want to point out is that in our mechanism each state transition can be triggered by any one from a serial of signalling packets, which makes the mechanism robust to potential packet loss due to congestion or channel error.

C. Sender Algorithm

The algorithm at the sender side is very simple. The sender checks if the feedback packet also contains a *HOn*. If yes, it resets the RTT record with the new RTT sample measured on this feedback packet and marks all the following data packets as echoes to the receiver, unless the receiver explicitly wants to suppress these echoes; otherwise it just act in the normal way. Following is the pseudo code of the sender algorithm:

```

feedback packet received
if with handover notification then
    reset RTT record
    if acknowledgements suppressed then
        do not mark following data packets
    else
        mark following data packets
    end if
else
    do not mark following data packets
end if
update RTT
other normal TFRC processes

```

V. PERFORMANCE EVALUATION

To show the improvement of our mechanism, we first generate the same set of figures as in Fig.1 for a directly comparison. The performance is then evaluated in different handover scenarios with various RTT pairs.

Fig. 4. Simulation Settings

Fig. 5. Fast Link Adaptation during a Handover with changed RTT

A. Simulation Setup

The topology in Fig.4 is used in the simulations. CN keeps sending a TFRC flow to the MN, while the latter switches from BS1 to BS2 through interface If1 and If2 respectively. The different latency between BS1-If1 and BS2-If2 changes the RTT experienced by the TFRC. The 3rd-layer lossless handover is realized in the way of having both interface active. Packets arriving at If1 after the handover are sent through If1-If2 to get marked. The handover happens at 30sec and the simulation lasts for 60sec.

B. Simulation Results and Analysis

The simulation results of the proposed mechanism are shown in Fig.5. Compared with Fig.1(b) and 1(d), both RTT and p in our mechanism are updated almost instantly after a handover. The effect on throughput and packet loss can be

Fig. 6. TFRC Receiver State Transition Graph

seen in Fig.5(a) and 5(c). There is almost no throughput drop any more in Case 1, and the packet loss in Case 2 is also greatly reduced (from 114 packets to only 19 packets). Please note that the negative peak in X_{rcv} in Fig.5(a) and 5(c) is not the real receiving rate but an artificial value to regulate the sending rate, as we discussed before.

Now we keep the delay in BS2-If2 always at 25ms and change the delay in BS1-If1 from 5ms to 70ms. The number of packet loss and the average throughput over 1sec after the handover are shown in Fig.6. It is clear that our mechanism has a consistent performance in all the different scenarios. The original TFRC, on the other hand, has lower throughput when the RTT increases; when the RTT decreases, it achieves a bit higher throughput at the cost of heavy packet loss.

VI. CONCLUSION

In this paper we studied the effect of changed RTT on TFRC handovers. A fast link adaptation mechanism is proposed for TFRC to adjust quickly to the RTT on the new link. During a handover the sending rate is simply kept around a desired value; as soon as packets arrive via the new link, TFRC is set directly to a state adjusted according to the sending rate and the new RTT. The proposed mechanism indeed improves the TFRC performance when RTT is changed due to a handover. It can work in conjunction with other bandwidth adaptation techniques to handle more general handover cases.

REFERENCES

- [1] S. Floyd, M. Handley, J. Padhye and J. Widmer, *TCP Friendly Rate Control (TFRC): Protocol Specification*, RFC Draft 04. IETF, Jan. 2008.
- [2] S. Floyd, M. Handley, J. Padhye and J. Widmer, *Equation-Based Congestion Control for Unicast Applications*, Proc. ACM SIGCOMM 2000, pp. 43-56, Stockholm, Sweden, August 2000.
- [3] J. Padhye, V. Firoiu, D. Towsley and J. Kurose, *Modeling TCP Throughput: A Simple Model and its Empirical Validation*, Proc. ACM SIGCOMM 1998, pp. 303-314.
- [4] A. Gurtov and J. Korhonen, *Measurement and Analysis of TCP-Friendly Rate Control for Vertical Handovers*, ACM MCCR 8(3), pp. 73-87, July 2004.
- [5] Y. Pan, M. Lee, J. B. Kim and T. Suda, *Smooth Handoff Scheme for Stream Media with Bandwidth Disparity in Wireless Cells*, Proc. CCW 2003, pp. 9-16, Oct. 2003.
- [6] Y. Lin, G. Kuo, H. Wu, Y. Peng and S. Cheng, *MBTFRC: A TFRC Enhancement for Heterogeneous Mobile Networks*, Proc. GLOBECOM 2003, vol.5, pp.2880-2884, Dec. 2003.
- [7] D. Li, K. Sleurs, E. Van Lil, A. Van de Capelle, *Improving TFRC Performance against Bandwidth Change during Handovers*, Proc. IEEE Wicomm 2008, Oct. 2008.