Rate Control Adaptation for Heterogeneous Handovers

Olivier Mehani,[1,⋆] Roksana Boreli[1,2] Guillaume Jourjon,[1] and Thierry Ernst[3,4]

⋆Corresponding author: olivier.mehani@nicta.com.au
[1]Nicta, Sydney. Eveleigh, NSW, Australia, first.last@nicta.com.au
[2]University of New South Wales, Sydney, Australia
[3]Mines ParisTech, Paris, France
[4]Inria, Rocquencourt, Paris, France, thierry.ernst@inria.fr

ISSN: 1833-9646-6163

NICTA Technical Report TR-6163

Rate Control Adaptation for Heterogeneous Handovers

Olivier Mehani, Roksana Boreli Guillaume Jourjon, and Thierry Ernst

Abstract:

We present enhancements to the TCP-Friendly Rate Control mechanism (TFRC) designed to better handle the intermittent connectivity occurring in mobility situations. Our aim is to quickly adapt to new network conditions and better support real-time applications for which the user-perceived quality depends on the immediate transmission rate. We propose to suspend the transmission before disconnections occur, in a way inspired by Freeze-TCP, and extend the solution by probing the network after reconnecting to enable full use of the newly available capacity.

We first introduce a numerical model of TFRC's performance after a network handover and use it to evaluate the potential performance gains for realistic network parameters. We then describe a set of additions to TFRC to achieve these gains. Implementations within the Datagram Congestion Control Protocol (DCCP) for *ns*-2 and Linux have been adapted to support these enhancements. Comparisons of experimental results for the original and modified DCCP are presented for a number of example mobility scenarios.

We thus show how the proposed modifications enable faster recovery after disconnected periods as well as significantly improved adjustments to the newly available network conditions and an improvement in the quality of experience (QoE) for video-streaming applications.

Keywords: TFRC, congestion control, vertical handover, cross-layer, DCCP, transport protocol

1 Introduction

In recent years, there has been a shift towards increased use of mobile devices for Internet access and the use of real-time applications such as multimedia streaming, VoIP and video conferencing in mobile environments. This has been in line with the increased capacity and widespread coverage of wireless technologies, primarily cellular mobile (3–4G) and Wi-Fi. The vast majority of today's mobile devices, in fact, include both 3G and Wi-Fi access options. Mobility support for moving between either the same technology (horizontal handovers) or cross-technology networks (vertical handovers) is supported by technologies such as Mobile IP [1]. However, the supporting transport protocols used for the emerging applications are still those which carried traffic for wired devices.

In this work, we consider *break-before-make* (also sometimes called *hard*) handovers between networks with heterogeneous characteristics. In our generic scenario, depicted in Figure 1, a wireless mobile node (MN) moves between two or more networks while having established sessions with fixed correspondent nodes (CN) in the Internet. The handover can be between two access points of the same technology, that is, *horizontal*, or *vertical*, between heterogeneous wireless networks such as 3G to Wi-Fi. Even though network mobility schemes hide most of the complexity of roaming from the upper layers, cross-technology handovers usually result in short, predictable disconnections during which network packets are bound to be lost. As most available congestion control mechanisms interpret such losses as an indication of congestion, they do not adapt well [2].

UDP is currently often used to carry real-time traffic. As it does not provide congestion control, it is not subject to this type of problem. However, it is argued in [3] that congestion control mechanisms should always be used in a shared internet. TCP is not a viable option for this type of content as retransmitted packets may already have gone past their usefulness deadline by the time they reach the receiver. To bridge this gap, the Datagram Congestion Control Protocol (DCCP) [4], [5] has been proposed as a non-reliable but congestion-aware transport protocol. DCCP can make use of the TCP-Friendly Rate Control mechanism (TFRC) [6]–[8], which replicates TCP's response to adjust to the network capacity. As it mimics TCP's congestion control response, TFRC experiences similar problems [9].

In this article, we study the behaviour of TFRC over heterogeneous handovers and propose mobility-aware enhancements. We first analytically model the effects of handovers on TFRC in order to quantitatively derive the potential for improvement. This model details a complete handover and allows to evaluate the number of lost packets during the disconnection to estimate the under-usage of the newly available network capacity after the reconnection. It is validated through *ns*-2 simulations. We then present an end-to-end solution to better cope with such events. This solution results in the extension of the TFRC protocol which we implement and evaluate both in *ns*-2 simulations, and experiments using our Linux code.

Some other solutions to the handover problem have been proposed in the literature. However, to the best of our knowledge, no fully integrated solution handles changes in the path characteristics after a "lossy" handover. Our disconnection-tolerant modification of TFRC shares similarities in concept with Freeze-TCP [10] but has different target applications and introduces further enhancements. This solution also relies on explicit handover notifications (in a way similar to [11] on link triggers) to be informed of mobility events, but the end-to-end path characteristics are then discovered uniquely at the transport layer.

The main benefits of our proposal include preventing unnecessary rate reductions by the transport protocol, both by avoiding false congestion detection and adapting faster

to the newly available capacity after vertical or horizontal handovers. The proposal also enables either side of the connection to suspend traffic entirely for the duration of the handovers. Thus, the resulting Freeze-DCCP/TFRC is a disconnection-tolerant congestion-controlled protocol for datagrams. The use of standard header fields and options makes our proposal backward compatible with existing implementations of DCCP.

Our Linux implementation allows us to show that this solution is well suited for real-time traffic while coping with heterogeneous mobile handovers with predictable disconnections. In particular, we focus on the quality of experience (QoE) [12] in the form of PSNR [13] for video streaming. This performance evaluation has been done over an OMF-enabled testbed [14], allowing for reproducibility and verification of the experiments.

This article is structured as follows: Section 2 presents some related work and background information. Section 3 uses simulations to highlight the impact of mobility-induced disconnections on TFRC, then introduces and validates a numerical model of its behaviour, allowing to estimate possible performance gains. Section 4 presents the proposed TFRC protocol modifications and their implementation into Freeze-DCCP/TFRC; *ns*-2 simulations and experimental results with a Linux implementation are presented in section 5. Finally, in Section 6, we summarise this work and present future research directions.

2 Related Work

The research literature contains a large number of contributions to deal with node mobility, at all the layers of the TCP/IP stack [15], as well as discussion about their validity [16]. At the transport layer, the proposals address the problem of sessions surviving the change of address of the endpoints. Though some generic work address wireless issues, only a few works focus on adaptation to inherently heterogeneous paths in vertical mobility handovers. Here, we remind the reader to the state of the art of congestion control mechanisms for the Internet. We also summarise the problems which arise in mobile and wireless environments, and review the proposed mitigation techniques.

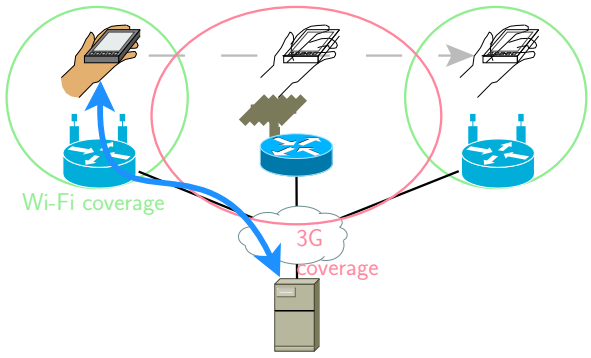


Figure 1: Generic use-case scenario: vertical hand overs between a number of access networks with different characteristics.

2.1 Congestion-Controlled Transport Protocols

It is important to fairly share the resource. It is therefore recommended to always use congestion-controlled transport protocols [3]. TCP is the epitome of such mechanisms. Its standard mechanism [17], [18] (comprising a slow-start and a congestion avoidance phase based on AIMD) is most notably based on a *congestion window* (`cwnd`) tracking the number of bytes that can be in flight and a *receiver window* (`rwnd`) manipulated by the receiver to provide flow control. TCP's congestion control is currently the standard of fair sharing on the Internet [3], [19]. There is however some contention with respect to how this fairness should be assessed [20].

The Stream Control Transmission Protocol (SCTP) [21] has been designed to offer a more flexible choice of feature combinations than the all-or-nothing dichotomy of TCP and UDP. It implements an AIMD-based congestion control similarly to TCP, but provides transport for datagrams rather than byte streams. One of its salient features is its capability to register multiple address for the session's endpoints to provide a failover mechanism if the primary path were to fail. It has been leveraged for handovers management [22], [23].

The Datagram Congestion Control Protocol (DCCP) [4], [5] has been proposed to provide congestion-controlled datagrams streams. However, contrary to SCTP, it does not enforce delivery reliability. This makes it a well suited transport for application such as real-time multimedia streaming, where timeliness of data arrival is more important than reliability. DCCP has been designed with modularity in mind, and several congestion control mechanisms (identified by their *Congestion Control Identifier*, CCID) can be chosen. A TCP-like congestion control algorithm, CCID 2 [24], follows TCP's `cwnd`-based mechanism made of slow-start and AIMD. CCID 3 [25] uses TFRC (see below). CCID 4 [26] has also been proposed as a variant of CCID 3 for small packets such as used for VoIP.

The TCP-Friendly Rate Control (TFRC) is not a transport protocol *per se*, but an equation-based rate control mechanism [6]–[8]. It uses network-gathered metrics like RTTs, number of lost packets (more precisely, the *loss event rate*) and the data rate observed by the receiver to compute the allowed sending rate, following a model equation of the throughput of TCP Reno under the same conditions [27]. Its operation is described in more details in Section 3.1. The use of an equation-based rate control makes the rate changes smoother, which is more appropriate for the streaming of multimedia content than the abrupt changes introduced by AIMD [8]. It has also been argued in [28] that such class of rate controls tends to be more resilient to wireless losses.

2.2 Wireless Issues and Proposed Solutions

In a well maintained wired-only network, the *only* possible source of losses is a router dropping packets due to its queue being full, that is, a congestion. In this context, packet losses can clearly be considered to be fully equivalent to congestion events, as the aforementioned congestion control mechanisms assume. Wireless links, however, can experience losses for other reasons such as those related to propagation impairments or collisions at the receiver.

TCP's AIMD does not handle such losses well, as it interprets them as signs of congestion. New update laws for `cwnd` have therefore been proposed to overcome this problem. TCP Westwood [29] and Westwood+ [30] estimate the end-to-end capacity based on the rate of acknowledgements (ACKs) and adapt the `cwnd` to match this estimate.

It is noted in [31] that multiple layer-2 mitigating solutions have also been proposed. Some standardised MAC protocols implement mechanisms to remediate or even avoid these losses. For example *Request to send/Clear to send* (RTS/CTS) mechanisms can be used to reserve the channel between both nodes and mitigate the hidden node problem. Most MAC mechanisms also adjust the physical data rate depending on the channel conditions to ensure the majority of the packets can be successfully received.

These MAC techniques are however not entirely transparent to the upper layers and, if they successfully recover from a link-layer loss, it is at the price of an increased delay to transmit the packet or an overall rate reduction. Several studies have confirmed the performance degradation of TCP on these wireless media [32]–[35].

To alleviate the impact of serious degradations of the wireless channel on the TCP congestion control, an enhancement named Freeze-TCP [10] attempts to detect such situations at the receiver and to suspend temporarily ("freeze") the sender. To this end, it leverages the window-based flow control mechanism of TCP by advertising a null `rwnd` (*zero-window advertisement*) when the wireless channel fades away. When the channel is restored to a usable level, the receiver sends a non zero-window advertisement, therefore resuming the sender's operation with the same congestion window. It was extended to address predictable fadings in vehicular networks [36] or mitigate the impact of case of vertical handovers [37]

As for TCP, losses due to contention in wireless LANs disrupt TFRC's rate estimation [38]. The authors of these works identify the specific case where TFRC's natural rate increase during loss-less periods leads to the wireless medium being saturated. The thus-created losses lead TFRC to reduce its rate, and eventually results in an oscillating behaviour. The authors therefore suggest to limit the transport protocol's sending rate control law to the data rate currently achievable by the underlying wireless link. They further extend the proposal by adding a similar constraint in order to fairly share the wireless link with other users.

2.3 Adapatability to Changes of Network Characteristics

Congestion control mechanisms usually rely on estimates of the network characteristics aggregated over time. In vertical handovers, it is not unlikely that the new network characteristics are very different from the previous one's. It will however take some time for the internal estimates to converge to the new values, and properly adapt the rate.

To address this problem, SCTP is extended, in [39], with a packet-pair probing of the failover path prior to switching traffic. Similarly, MBTFRC [40] as well as the work in [41], have been proposed to implicitely identify changes in the network characteristics, and react to them adequately. Packet-pair techniques are however questionned as to their ability to adequately estimate the capacity of a loaded path [42]. High-to-low capacity changes are addressed in [43] which proposes some self-clocking *à la* TCP to avoid overloading a slow or congested link. A more aggressive extension is also proposed in [44]. To accomodate for intermitent traffic without unduly limiting the rate, a faster restart has also been proposed for TFRC [45], which probes the network more aggressively under the assumption that its condition might not have changes much since the last estimate.

Our proposal differs from the approaches above in that it is an integrated transport-level solution which addresses both high-to-low and low-to-high handovers and adjusts the rate accordingly through in-band techniques. Moreover, it also targets break-before-make events (of which *make-before-breaks* are a subset) and the resulting packet losses during the disconnected period. Finally, it relies on explicit notifications (the only cross-layer information needed), which allows for a more timely and accurate knowledge of path changes.

3 Behaviour of TFRC During a Disconnection

In this section, we investigate the issues TFRC faces when used in mobile scenarios with heterogeneous handovers. We first describe the operation of TFRC as standardised by [8]. We then provide an example simulation highlighting some of the issues. The behaviour illustrated here is then modelled in order to quantify the performance issues.

3.1 Operation of Standard TFRC

Based on feedback from the receiver, a standard TFRC sender controls its rate X following a model of TCP's throughput under the same conditions [27],

$$X_{\mathrm{Bps}} = T(p,R) = \frac{s}{R\sqrt{\frac{2p}{3}} + t_{RTO}\left(3\sqrt{\frac{3p}{8}}\right)p\left(1+32p^2\right)}, \tag{1}$$

$$X \leftarrow \min(X_{\mathrm{Bps}}, 2X_{\mathrm{recv}}) \tag{2}$$

where s is the packet size, R the RTT, and t_{RTO} the retransmit timeout (usually $4R$, measured by the `nofeedback` timer). Parameters p and X_{recv} are reported by the receiver roughly every RTT and are, respectively, the loss event rate and the current received rate. If no report from the receiver is seen before t_{RTO} expires, the sender reduces its allowed sending rate by halving the last used X_{recv}.

As for TCP, a slow-start phase is also present to first adapt the rate to the network path's capacity. During this phase, the sender updates its rate once per RTT following

$$X \leftarrow \min(2X, 2X_{\mathrm{recv}}), \tag{3}$$

until the first loss is observed. When the first loss occurs, the TFRC receiver reports a loss event rate p which reflects its observed throughput X_{recv} before the loss. The value of p is initialised by inverting (1).[1]

[1] It is not specified in [8] how this inversion should be done. Most implementations use a binary search, but [46] suggests a more CPU efficient method based on a Newton search.

When further losses are observed, the TFRC receiver computes p as the inverse of the weighted average of the lengths of the n most recent loss intervals $i_0, \ldots, i_{n-1}$. The length of a loss interval is measured in number of packets successfully received. The average is computed using a vector of decreasing weights $\vec{w} = [w_0, \ldots, w_{n-1}]$ as $S_0 = \sum_{i=0}^{n-1} w_i i_i$. The TFRC receiver actually keeps a history $\vec{i} = [i_0, \ldots, i_n]$ of the last $n+1$ loss intervals. This slightly larger buffer is designed to avoid overly increasing the loss event rate when one of these events has just happened. Indeed, when losses have just been experienced, the size of the current loss interval i_0 starts increasing from 0. At first, i_0 is so small that it would incorrectly drive p up and needlessly reduce the rate $X_{\text{Bps}} = T(p, R)$. It is therefore ignored and the reported loss event rate is still based on the previous $i_1, \ldots, i_n$ intervals. As these values do not change anymore, p is stationary during this period. Taking $S_1 = \sum_{i=0}^{n-1} w_i i_{i+1}$, p is computed in [8] as

$$p = \frac{1}{i_{\text{mean}}} = \frac{\sum_{i=0}^{n-1} w_i}{\max(S_0, S_1)}. \quad (4)$$

Computing p this way only considers the duration of lossless periods and is not related to that of those during which all packets are lost, even if they span several RTTs.

As (1) has an inverse relation with the loss event rate p, TFRC is not fit to work on networks with loss-inducing disconnections. A temporary break in the end-to-end path would indeed have several consequences. First, packets will needlessly use parts of the network path's capacity before being dropped, resulting in losses. The sending rate will then gradually be reduced as the `nofeedback` timer expires. Upon reconnection, the transport protocol's rate will therefore not match the network's characteristics and need some time to re-adapt. We illustrate this behaviour in the next section.

Moreover, as the computation of the loss event rate is based on a history of several loss events, TFRC reacts slowly to immediate decreases in p. In the case of a cross-technology hand-off to an access networks with larger capacity, it will therefore take a much longer time to adjust the rate to the newly available capacity.

3.2 Simulation of Mobile Handovers

To present an example of the adverse consequences of disconnections on the sending rate of DCCP/TFRC in mobility situations, we ran several simulations with *ns*-2 [47]. The simulation scenario consists of a landscape of 800 × 1600 m where a Mobile IPv6 (MIPv6, [48]) mobile node (MN) moves between the coverage of three access routers (AR). Figure 2 presents the simulated environment, consisting of one backbone router and the three ARs providing adjacent but non-overlapping wireless connectivity to the MN receiving traffic. Simulations were also run with the second base station disabled in order to observe the behaviour of the data stream in the case of a more sporadically available network coverage.

ns-2 was configured to simulate a regular single-rate 11 Mbps 802.11b wireless channel. Some parameters had to be adjusted to obtain the desired simulation conditions: the wireless reception threshold has been fine-tuned to simulate

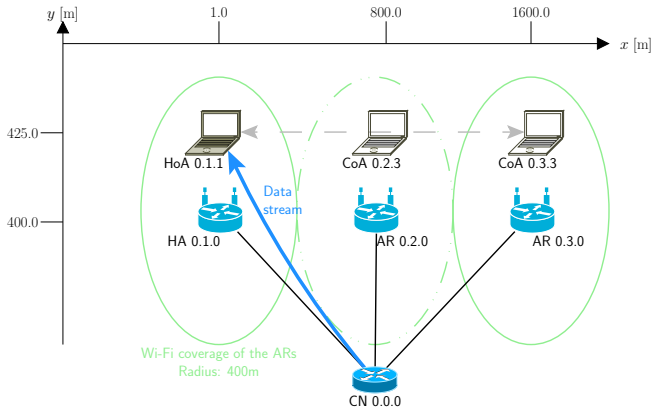


Figure 2: The basic simulation scenario. A mobile node (MN) moves back and forth between three adjacent (but not overlapping) wireless networks with different prefixes. The correspondent node (CN) sends a constant stream of data to the MN's home address (HoA) using DCCP/TFRC. For simplicity, the MN's home agent (HA) is set to be the access router (AR) of the first access network. Addresses are expressed in *ns*-2 format.

Table 1: Parameters adjusted from *ns*-2's defaults.

***ns*-2 parameter**	**Value**
11 Mbps 802.11b channel	
`Phy/WirelessPhy bandwidth_`	`11Mb`
`Phy/WirelessPhy freq_`	`2.472e9`
`Mac/802_11 dataRate_`	`11Mb`
`Mac/802_11 basicRate_`	`1Mb`
Miscellaneous	
`Phy/WirelessPhy RXThresh_`	`5.57346e-11`
`Agent/MIPv6/MN bs_forwarding_`	`0`
`Agent/MIPv6/MN rt_opti_`	`0`

a 400 m Wi-Fi range. Table 1 summarises these configuration changes. We have also ported the MobiWan MIPv6 support [49] and DCCP module [50] to version 2.33 of this simulator, and updated these to the latest versions (at the time) of their respective specifications [8], [48].[2] All kinds of route optimisations for MIPv6 were disabled.

In the first scenario, the MN moves back and forth at constant speed between all three ARs (from adjacent Wi-Fi networks), losing connectivity with the current one, associating with the new one, and re-establishing its mobility bindings with its HA. Figure 3 shows that, in addition to the delay to associate with the new AR and re-establish the bindings when the previous link breaks, there is a delay between the time when a care-of address (CoA, the locator in the current access network) is fully configured on the new access network, and when the rate of TFRC is reinstated: 100 ms until it restarts, but 500 ms until it is fully restored. Figure 4 shows the results for a similar scenario where the second access point has been disabled, thus creating a period of complete lack of connectivity. The previous delay effect becomes much larger, up to 50 s in this case.

In the next section, we model this behaviour in order to

[2] These updated patch-sets are available at http://www.nicta.com.au/people/mehanio/nsmisc/.

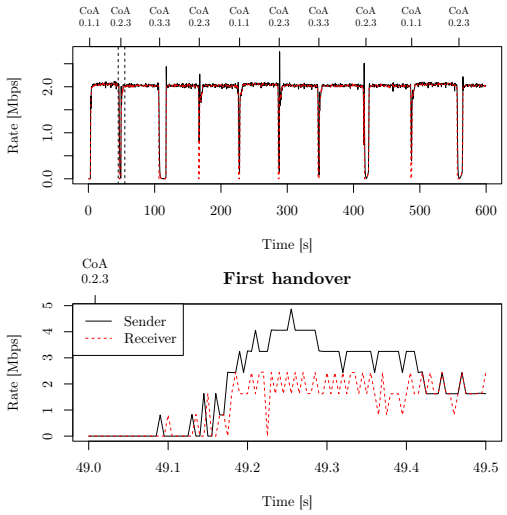


Figure 3: DCCP/TFRC data flow moving through adjacent Wi-Fi access networks. Labels on the top axis represent the time when the new CoA has been configured and is fully usable.

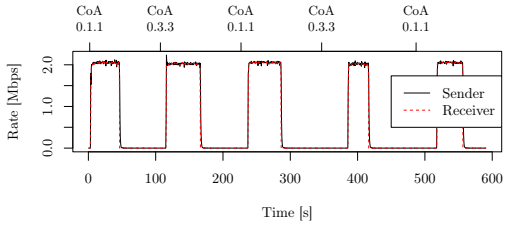


Figure 4: DCCP/TFRC data flow moving through non-adjacent Wi-Fi access networks.

evaluate the performance issues in terms of lost packets, delay until restart and "wasted" capacity.

3.3 Numerical Model of TFRC's Behaviour

In order to confirm the generality of the example of the previous section, and quantify the highlighted impact, we introduce a model of TFRC's behaviour when a disconnection occurs. This model is used to derive the number of packets that are lost during the disconnection, the delay before TFRC resumes sending after a reconnection, the available capacity and the time it takes to adapt to the new characteristics. After validating it with *ns*-2 simulations, we evaluate these metrics for various typical horizontal and vertical handover scenarios. This shows that there is ample room for better management of these events. A summary of the symbols used throughout this section is given in Table 2.

3.3.1 During the Disconnection

During the disconnection, the TFRC senders keeps transmitting packets. However, no feedback from the receiver can be received, and the sending rate is gradually reduced. Here, we evaluate this rate, and the number of packets which get sent but cannot be received.

Evolution of Internal Parameters We evaluate the changes in sender rate X during a disconnection, as well as the `nofeedback` timer's period, t_{RTO}. Both values have an impact on the number of packets lost during the disconnection and the rate recovery after the reconnection. Figure 5a represents the evolution of these parameters.

Table 2: Notations used for the analysis of TFRC.

Sym.	Meaning
	During disconnection
t^i_{RTO}	Duration of NFI i
X^i	Sender rate during NFI i ($X_d = X^0$)
i_x	First NFI so that $X^{i_x} = s/t_{\text{mbi}}$
i_t	First NFI so that $t^{i_t}_{\text{RTO}}$ starts increasing
n_{lost}	Number of packets lost during the disconnection
	After reconnection
t_{idle}	Time before the first packet is sent after reconnection
n^{ε}_R	Number of RTTs on the new network before R^i_r is within ε of R_{new}
n^i_{pkts}	Total number of packets sent after RTT i on the new network
n^*_{wasted}	Number of packets that could have been sent

Just before the disconnection occurs, at T_d, the sender sends at rate $X = X_d$. In the following, this is assumed to be the nominal TFRC rate that the underlying link can support. Consequently, the receiver measures and reports an X_{recv} roughly equal to X_d. Thus, (2) is limited by X_{Bps}.

In the absence of feedback, the TFRC sender halves its allowed sending rate every time the `nofeedback` timer expires by reducing its local estimate of X_{recv}. When X becomes small, t_{RTO} is increased to cover the transmission of at least two packets.

For convenience, we segment the disconnected period into *no feedback intervals* (NFI). An NFI is the interval between two consecutive expirations of the `nofeedback` timer.[3] NFIs are indexed starting at $i = 0$. The first expiration of the `nofeedback` timer marks the end of NFI 0. Hence, the effects of this timeout start at the beginning of NFI 1. The rate then gradually decreases until it reaches its minimum value during NFI i_x.

Every NFI, the sender halves the value of X_{recv}, which then drives (2). In the worst situation, X can reduce to the minimal value of one packet every 64 seconds (s/t_{mbi}). Taking i_x as the NFI during which $2X^{i_x}_{\text{recv}}$ drops below s/t_{mbi}, the sender rate can be expressed as

$$X^i = \begin{cases} \frac{X_d}{2^i} & \text{if } 0 \le i < i_x, \\ \frac{s}{t_{\text{mbi}}} & \text{otherwise,} \end{cases} \quad \text{with } i_x = \left\lceil \log_2 \frac{X_d \cdot t_{\text{mbi}}}{s} \right\rceil, \tag{5}$$

where $\lceil \cdot \rceil$ is the ceiling operator.

Additionally, the `nofeedback` timer, initially set to $t^0_{\text{RTO}} = 4R$, increases when the sending rate becomes smaller than $2s/4R$. Assuming $X_d \ge 2s/4R$ and taking i_t as the NFI during which $2s/X^{i_t}$ becomes larger than $4R$,

[3] An NFI is the same concept as the NFT of [45].

the duration of NFI i is then

$$t^i_{\text{RTO}} = \begin{cases} 4R & \text{if } i < i_t, \\ \frac{2s}{X^i} & \text{otherwise,} \end{cases} \quad \text{with } i_t = \left\lceil \log_2 \frac{2R \cdot X_d}{s} \right\rceil. \quad (6)$$

Note that (6) is only valid for $R < t_{\text{mbi}}/2$, in which case $i_t \leq i_x$. Otherwise, $4R$ is larger than the time to send 2 packets at the lowest rate, and i_t is considered to be $+\infty$.

Packet Losses Though the number of losses happening during the single loss event of the handover does not directly impact TFRC's sender rate, they are an unnecessary burden on the rest of the network which could be better used for other traffic, for which data can actually be delivered to the destination. It is interesting to quantify this charge on the network in the form of the number of packets which will be lost during the disconnected period.

Figure 5b shows the evolution of the sender rate during a handover. Two cases are represented, for different reconnection times T_c and T'_c. They respectively occur before and after the sender's estimation of the receiver rate has reduced to less than one packet per RTT.

Time $t_D = T_c - T_d$ is the length of the disconnected period. All the packets sent during this period are lost. The number of lost packets when the reconnection occurs, after n_D NFI (such that $\sum_{i=0}^{n_D} t^i_{\text{RTO}} \geq t_D$), can be estimated using

$$n_{\text{lost}} = \begin{cases} \left\lfloor \frac{t_D X^0}{s} \right\rfloor & \text{if } t_D \leq t^0_{\text{RTO}}, \\ \left\lfloor \frac{t^0_{\text{RTO}} X^0}{s} + \sum_{i=1}^{i_D} \frac{t^i_{\text{RTO}} X^i}{s} \right\rfloor & \text{otherwise,} \end{cases} \quad (7)$$

where $i_D = n_D - 1$ is the index of the n_D^{th} NFI and $\lfloor \cdot \rfloor$ is the floor operator.

3.3.2 After the Reconnection

After having reconnected, packets and feedback messages can be exchanged again, but some waiting time due to exponential backoff may delay this restart. The loss event rate p gets updated on the first feedback message, then so is the rate. When the rate can increase again, it will however reflect the parameters observed on the previous network path, which may be inappropriate for the new access network.

Variation of the Loss Event Rate The losses will only be noticed by the receiver after reconnecting. In [8], the authors specify that the expected arrival time of a lost packet is interpolated using those of both packets received directly before and after the loss. Multiple losses over the disconnected period will then be considered part of the same loss event starting in the middle of the disconnected period.

Following the procedure described in Section 3.1, the evolution of p can be in three different phases depending on S_0 and S_1 (defined in that section):

no loss when $S_0 \geq S_1$, p gradually decreases as the number of received packets, in i_0, increases;

first loss observed makes $S_0 < S_1$ which stabilises p until the current loss interval i_0 becomes large enough that the inequality is reversed;

new loss observed when $S_0 < S_1$, $\vec{i}$ gets shifted which increases p as (4) now has a smaller denominator.

In the congestion avoidance state, the variation of loss event rate (Δp) can be in different ranges depending which of the three phases the sender is in.

$$\begin{cases} \Delta p = 0, & \text{after the first loss has been observed,} \\ \Delta p > 0, & \text{when more losses are observed,} \\ \Delta p_{\min}(\Delta n_{\text{pkts}}, p_{\text{prev}}) \leq \Delta p < 0, & \text{otherwise (no loss).} \end{cases} \quad (8)$$

The lower bound of $\Delta p = p - p_{\text{prev}}$ when no losses occur can be derived using (4) to estimate p. If all Δn_{pkts} packets sent since the last feedback, which reported p_{prev}, have been received and taken into account in the receiver's next feedback, the reduction in p will be the lower-bound

$$\Delta p_{\min}(\Delta n_{\text{pkts}}, p_{\text{prev}}) = \underbrace{\frac{\sum_{i=0}^{n-1} w_i}{w_0 \Delta n_{\text{pkts}} + (\sum_{i=0}^{n-1} w_i)/p_{\text{prev}}}}_{p = 1/i_{\text{mean}}} - p_{\text{prev}}. \quad (9)$$

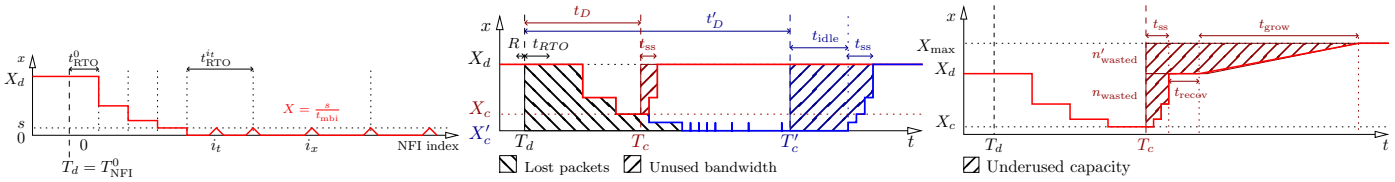
Event Timing While the time base of changes of the disconnected sender are regulated by the length of its retransmit timeout, feedback from the receiver takes on this role once reconnected. The receiver sends periodic feedback messages at least once per RTT. For the rest of this section, time will then be segmented in units of RTTs. Indices i now refer to how many of those have elapsed since the reconnection, rather than NFI as previously.

The periodicity of sender-side events triggered by these feedback messages will follow the RTT of the visited network. While this value is mostly stationary for a given network, the sender does not use it directly for its computations, most notably that of the sending rate. To ensure a smooth evolution, it uses an exponentially weighted moving average of the past samples to estimate the RTT. After feedback message i allowing to sample RTT R_{new} of the new network, the sender's estimate of the RTT is $R^i_r = qR^{i-1}_r + (1-q)R_{\text{new}}$, with $0 < q < 1$ ($q = 0.9$ in [8]). Just after the reconnection, before the first feedback message has been received, the estimate completely reflects the RTT of the previous network, $R^0_r = R_{\text{old}}$. Expanding the series, this estimate can be expressed as

$$R^i_r = (1-q)R_{\text{new}} \underbrace{\sum_{j=0}^{i-1} q^j}_{\frac{q^0 - q^i}{1-q}} + q^i R^0_r = \left(1 - q^i\right) R_{\text{new}} + q^i R_{\text{old}}. \quad (10)$$

It will then evolve from a value representing of the RTT on the previous network, R_{old}, to that of the new network. Depending on the difference ratio between these two values, a variable number of samples will be needed for the sender to have an accurate estimate of R_{new}. The number of samples, denoted n^ε_R, needed to have an estimate within ε of the actual value, that is, $\left| R^{n^\varepsilon_R}_r - R_{\text{new}} \right| \leq \varepsilon$, is

$$n^\varepsilon_R = \left\lceil \frac{\ln \varepsilon - \ln |R_{\text{old}} - R_{\text{new}}|}{\ln q} \right\rceil. \quad (11)$$



(a) Internal parameters (b) Rate behaviour (c) Larger capacity

Figure 5: TFRC sender's internal parameters and rate behaviours. (a) Evolution of internal parameters after a disconnection. (b) Rate behaviour in a period with no connectivity. Two cases are shown, with different times of reconnection: at T_c, a period t_D has elapsed which is short enough that TFRC's rate did not reach its minimum (red) and at T'_c, when the time t'_D elapsed since the disconnection was sufficient for X to reduce to s/t_{mbi} (blue); an additional delay t_{idle} is present in the latter case before TFRC starts restoring its rate. (c) After a reconnection, TFRC does not adapt quickly to higher capacities. It slowly uses more capacity as p decreases.

Assuming that the order of magnitude can vary from the millisecond to the second depending on the network technology and load, it can take up to almost 30 RTTs on the new network for the estimate to be accurate within a 5 % margin.

It is important to note that the TFRC equation (1) is directly dependent on this estimate. Given a static $p = p_r$ after the reconnection, as per (8), successive samples of the new RTT will refine the estimate which will in turn impact X_{Bps}. As $t_{\text{RTO}} = 4R$, X_{Bps} can be expressed as a function of any of its previous values $T(p_r, R')$ and the associated RTT estimate R'

$$X^i_{\text{Bps}} = T(p_r, R^i_r) = \frac{R'}{R^i_r} T(p_r, R'). \tag{12}$$

The most useful such relation involves $X_d = T(p_r, R_{\text{old}})$: $X^i_{\text{Bps}} = (R_{\text{old}}/R^i_r)X_d$; upon reconnection, the new TFRC rate, as compared to that before the disconnection, is thus only dependent on the ratio of the current and previous estimations of the RTT, as p_r is computed from S_1.

Number of Sent Packets When the connection is re-established and the TFRC sender restarts sending packets, it goes through a few phases before being able to resume a rate appropriate to the current network. Depending on the differences between the previous and the current networks' characteristics, the duration (or existence) of these phases will vary. An important factor impacting theses phases is the number of packets that have been sent since the reconnection. In the following, it will be expressed as $n^i_{\text{pkts}} = 1/s \sum_{j=0}^{i} R^j_r X^j_r$, where i is the RTT at the end of which the packets are counted and X^i_r is the sending rate during that RTT, as detailed below.

Unused Capacity When connectivity is re-established, two factors can prevent the TFRC sender from fully utilising the available capacity instantaneously. First, when feedback is received, the sending rate is not resumed directly, but through a phase similar to slow-start (*e.g.*, at T_c in Figure 5b). Second, the sender is not directly aware of the reappearance of connectivity. It has to wait for a packet to be acknowledged by the receiver. As the sending rate has been gradually reduced, said packet may not be immediately sent (*e.g.*, t_{idle} after T'_c in Figure 5b).

If the sending rate when reconnecting, $X_c = X^{n_D}$, is small, the delay s/X_c between the transmission of two subsequent packets becomes significant. When connectivity is recovered, it can take up to this delay before the first packet is sent. The average idle time after reconnecting can be expressed as $t_{\text{idle}} = s/2X_c$.

After this delay, the sender eventually starts increasing the rate. Packets are first sent at rate X_c. Every RTT, a feedback is received with the current value of X_{recv}. According to (2), the rate can then be updated to twice this value until X^i_r reaches the rate allowed by the TFRC equation, $X^i_{\text{Bps}} = (R_{\text{old}}/R^i_r)X_d$. During the slow-start RTT i, the sender rate is

$$X^i_r = 2^i X_c. \tag{13}$$

Assuming the new network provides the same capacity as the previous one, the average number of packets that could additionally be sent is

$$n_{\text{wasted}} = \frac{1}{s}\left(t_{\text{idle}} \cdot X_d + \sum_{i=0}^{n_{\text{ss}}} R_{\text{new}}\left(X_d - X^i_r\right)\right). \tag{14}$$

Parameter n_{ss} is such that $X^{n_{\text{ss}}}_r \geq X^{n_{\text{ss}}}_{\text{Bps}}$. The development of this inequality using (10), (12) and (13) leads to

$$\frac{R_{\text{new}}}{R_{\text{old}}} 2^{n_{ss}} + \left(1 - \frac{R_{\text{new}}}{R_{\text{old}}}\right)(2q)^{n_{ss}} > \frac{X_d}{X_c}, \tag{15}$$

which is not linear in n_{ss}. It thus cannot be solved in a purely analytical fashion. In our implementation of the model, we used the Newton-Raphson method [51] with

$$f(n_{ss}) = \frac{R_{\text{new}}}{R_{\text{old}}} 2^{n_{ss}} + \left(1 - \frac{R_{\text{new}}}{R_{\text{old}}}\right)(2q)^{n_{ss}} - \frac{X_d}{X_c} \quad \text{and} \tag{16}$$

$$f'(n_{ss}) = \frac{R_{\text{new}}}{R_{\text{old}}} 2^{n_{ss}} \ln 2 + \left(1 - \frac{R_{\text{new}}}{R_{\text{old}}}\right)(2q)^{n_{ss}} \ln 2q. \tag{17}$$

This allowed us to solve (15) for n_{ss} in only a few iterations starting from an arbitrary $n_{ss0} = 10$, regardless of the network parameters.

Networks with Larger Capacity The estimate of the loss event rate p is designed to evolve smoothly. This may cause an additional under-usage of the available capacity in case the MN connects to a network with a higher capacity, X_{max}, than the previous link. Depending on the difference

in capacity from said previous network, it may take an unacceptably long time for the sender to eventually occupy the full available capacity. Figure 5c shows how TFRC slowly adapts to the new network capacity.

This adaptation time can be split into two periods. First, once the slow-start phase has finished, the sender rate may not immediately start increasing above X_c. Indeed, if there has not been enough packets sent during the slow-start for S_0 to be larger than S_1 in (4), p will not decrease. During the loss recovery time, X_{Bps} is kept at value $X_r^i = (R_{\text{old}}/R_r^i)X_d$. After t_{recov}, when enough packets have been received, p will start decreasing again. During this phase, the sending rate slowly adapts to the available capacity, which is eventually reached after t_{grow}. In addition of n_{wasted}, a further n'_{wasted} packets could be sent. We develop a formulation of this extra capacity wastage below.

The loss recovery time, until the current loss interval contains enough packets not to be ignored in (4), is such that $0 < S_0 - S_1$ that is,

$$0 < w_0 \underbrace{i_0}_{n_{\text{pkts}}^{n_{\text{recov}}}} + \sum_{n=1,\ldots,i-1} (w_n - w_{n-1})i_n - w_{i-1}i_i. \quad (18)$$

To "compete in the global Internet with TCP," it is recommended in [8] to take $n = 8$ and the weight vector as $\vec{w} = [1, 1, 1, 1, 0.8, 0.6, 0.4, 0.2]$. The formulation of (18) can thus be simplified as

$$0 < n_{\text{pkts}}^{n_{\text{recov}}} - 0.2\sum_{n=4}^{8} i_n \quad \rightarrow \quad n_{\text{pkts}}^{n_{\text{recov}}} > 0.2\sum_{n=4}^{8} i_n. \quad (19)$$

When $n_{\text{pkts}}^{n_{\text{recov}}}$ packets have been sent since the reconnection, p, driven by (4), starts decreasing. It is difficult to estimate the i_n as they are dependent on the previous network conditions and specific history. However, assuming a relatively stable network, all i_n would be similar and close to the inverse of p_r, the loss event rate of the previous network. Thus, an estimate of the number of packets that need to be sent before p starts to adapt to the new network conditions can be written as $n_{\text{pkts}}^{n_{\text{recov}}} = 1/p_r$.

This estimation allows to evaluate the duration of the recovery period, t_{recov}, which exists only if $n_{\text{pkts}}^{n_{\text{recov}}} > n_{\text{pkts}}^{n_{\text{ss}}}$ (that is $t_{\text{recov}} > 0$) ,

$$t_{\text{recov}} = \frac{s}{Xd}\left(n_{\text{pkts}}^{n_{\text{recov}}} - n_{\text{pkts}}^{n_{\text{ss}}}\right) = \frac{s}{Xd}\left(1/p_r - n_{\text{pkts}}^{n_{\text{ss}}}\right). \quad (20)$$

The additional amount of wasted capacity can be estimated as

$$n'_{\text{wasted}} = \frac{1}{s}(X_{\max} - X_d)\left(t_{\text{idle}} + t_{\text{ss}} + t_{\text{recov}}\right) + \frac{R_{\text{new}}}{s}\sum_{i=0}^{n_{\text{grow}}}\left(X_{\max} - X_r^i\right) \quad (21)$$

with

$$X_r^i = \begin{cases} X_d & \text{if } i = 0, \\ \min\left(X_{\text{Bps}}\left(p_{\text{r}} + \Delta p(n_{\text{pkts}}^{i-1}, p_{\text{r}}), R_r^i\right), 2X_r^{i-1}\right) & \text{otherwise.} \end{cases} \quad (22)$$

Similarly to (14), n_{grow} is the number of RTTs needed to have $X^{n_{\text{grow}}} \geq X_{\max}$.

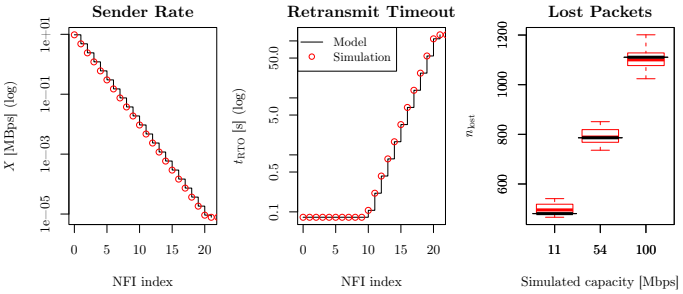


Figure 6: Comparison of simulated TFRC's internal parameters, and number of lost packets during disconnections with the model's predictions. Results shown for $X = 10\,\text{MBps}$ and $R = 2\,\text{ms}$.

3.4 Model Validation

We now verify the model of the previous section by comparing numerical results to *ns*-2 simulations for a wide range of network parameters and disconnection durations.

To check the behaviour of TFRC during the disconnected period—(5) and (6)—as well as the resulting number of lost packets (7), 60 s disconnections are introduced after a variable amount of time, for link capacities of 10, 54 and 100 Mbps and for a range of delays (1–100 ms). The number of packets sent after the disconnections is then counted in the simulation trace file. Figure 6 shows a comparison of simulation results with predictions from the model for $R = 1\,\text{ms}$. It confirms that our model exactly predicts the values of the internal parameters of the TFRC sender, and accurately estimates the number of losses.

The number of wasted packets, as determined by (14) and (21) cannot be derived in the same way from the *ns*-2 trace files. Detecting the end of the adaptation periods t_{ss} and t_{grow} relies on comparing the current rate to $X_{\max}$, for which it is impossible to obtain a ground truth from the simulations. It is therefore impossible to identify over which period to count the additional packets which could have potentially been sent, and the results vary depending on the estimate taken for $X_{\max}$. In the following, we take $X_{\max} = X_{\text{recv}}$ (from Table 3 below); the order of magnitude of the resulting numbers for the wasted capacity are coherent, but the reported value is only indicative, and should not be used in further derivations.

Even though the presented model does not encompass all the details of the behaviour of a real TFRC sender, it has proven to have sufficient prediction accuracy to be used in estimating potential performance gains.

3.5 Potential for Improvement

We use our numerical model to determine the performance improvements that can be expected from a better handling of disconnections. The input parameters (X_d and R of both links) for the model are those observed by TFRC when the stationary state has been reached. These are summarised in Table 3 (based on network characteristics presented later in Table 5).

MIPv6 relies on message exchanges over the network to update the binding with the HA.[4] Therefore, handover de-

[4] Fast handovers [52] could be used to reduce the disconnection

Table 3: Network parameters as observed by the *ns*-2 TFRC sender in the stationary phase, reached at T_{stat}.

Link type	X_{recv} **[MBps]**	R **[s]**	T_{stat} **[s]**
UMTS	0.044	0.96	660.54
802.16	1.10	0.17	264.14
802.11b	1.27	0.05	50.69
802.11g	4.82	0.04	21.67

Table 4: Predicted packet losses and wasted available capacity expected during a MIPv6 handover.

from \ **to**	UMTS	802.16	802.11 b	802.11 g
Packet losses				
UMTS	306	236	226	224
802.16	2760	2614	2614	2614
802.11b	1080	1078	1078	1078
802.11g	2909	2907	2907	2907
Wasted capacity [Number of 500 B packets]				
UMTS	0	8×10^4	3×10^2	1×10^5
802.16	0	5×10^2	2×10^2	1×10^3
802.11b	0	0	1×10^3	5×10^4
802.11g	0	0	0	5×10^3

lays will vary depending on the characteristics of the current link. The RTT on the new access network is particularly important as it affects the delay until the MIPv6 binding updates are received. In [53], the authors thus proposed to use $t_{\text{ho}} = 2.5 + R$ as the time to complete the handover, and during which packets cannot be successfully transmitted to the MN. We use this model as $t_D = t_{\text{ho}}$.

An estimate of the stationary phase value for RTT on the new link is used for R in t_{ho}. It is an over-estimation as the RTT is likely not to be as high upon reconnection as when the full rate is established. Therefore, the presented results should be considered an upper bound for the packet loss and lower bound for the wasted capacity.

The number of lost packets and the available capacity wasted during a MIPv6 handover, as predicted by the model, are shown on Table 4. These results confirm that the behaviour of TFRC can be improved. We intend to provide such improvement in the next section.

4 Freezing the DCCP/TFRC Transmission Upon Disconnections

In this section, we present an enhancement and its implementations to approach the possible improvement made apparent above. This modification relies on two main additional stages. The sender's state is first frozen just before a hand-off so as not to disrupt its performance, and transmission is suspended. When the handover is complete, the sender is unfrozen. Then, with assistance from the receiver it restores its previous rate and, if possible, probes the new network path for a larger usable capacity. Though this work primarily addresses break-before-make handovers, a subset can also apply to make-before-break events. In these cases, only the probing phase is needed.

The rationale for restoring the previous rate comes from Freeze-TCP [10], where only wireless fading or horizontal handovers were considered. Another advantage of this approach is that this rate is known without dependence any external information source. In the case of vertical hand-offs, as we consider them here, other approaches could also be envisionned. Some additional cross-layer interaction would allow the application or some management element to specify another restart point based on requirements or external knowledge of the expected capacity [*e.g.*, 54], [55].However, it would be important to ensure that the selected rate is not so high as to cause overly excessive congestion on the new path during the first RTT. In the following, we only consider previous-state restoration.

4.1 Rationale of the Improvements

Beyond the packet losses and under-usage of the available capacity, a reduction in the immediate rate is quite detrimental to real-time applications. As previously shown in Section 3.3.2, it can take up to several seconds to restore the rate after the completion of a handover. During this period, applications observe high error rates as they cannot fit the required amount of data units in the rate allowed by the transport protocol, which results in bad QoE. In such a situation, restarting from the rate achieved before the handover would enable the application to drastically reduce this period of bad quality.

Considering a communication involving video, the user's experience can be further improved if the new access network has better characteristics, such as a larger capacity. In the new network, the video codec could use a higher encoding rate which would increase the overall QoE. Including a mechanism to probe the new network path can make the new available path capacity available much faster to the application. It could then take advantage of this larger capacity to enhance the overall user experience.

Finally, having information about upcoming handovers, it is also possible to limit, or even nullify, the number of lost packets. Doing so also has the advantage of avoiding the unnecessary use of the old network path to send data which would never be received as the receiver has moved.

When a hand-off is imminent (as detected through , *e.g.*, IEEE 802.21 [56] or control frameworks [55]), we thus propose to temporarily suspend the evolution of specific internal parameters. The sender keeps an estimate of the current stationary parameters of the network used to derive the sending rate offered to the application. Further packet transmission is also prevented as the path from the sender to the receiver is known to be temporarily cut. When connectivity is available anew, the rate is restored immediately and adapted as soon as possible to the new network conditions. The congestion control algorithm first allows packets to be sent at the same rate as before. If no error is reported, the sender then tries to probe the network path by increas-

duration. However, packets arriving during the hand-off are buffered at the new access router, which is not desirable for real-time traffic as this would create latency at the application layer upon completion of the handover.

ing its rate. In a way similar to the initial slow-start, the sending rate doubles every RTT until the capacity of the new network has been reached.

4.2 New States to Support the Freezing Mechanism

We implement our proposed enhancement to TFRC within DCCP's CCID 3. The operation of the resulting Freeze-DCCP/TFRC is separated into three phases: *Frozen*, *Restoring* and *Probing*. New states are implemented into the TFRC sender and receiver to support these phases. Additionally, new DCCP options are introduced to enable the required freeze/unfreeze signalling, while state transition and synchronisation is done purely through TFRC options.

Figure 7 shows the proposed Freeze-DCCP/TFRC state diagram. The sender has three new states, shown in Figure 7a. As most of Freeze-DCCP's operation is driven by the sender, its states are directly named after the three phases. The receiver has two "active" states: *Restoration* and *Probed*. Both *Recovery* states are transient and used to ensure synchronisation. These are shown in Figure 7b.

The following sub-sections detail the signalling options and the evolution of the states, as well as their specific actions throughout the Freeze-DCCP/TFRC phases.

4.3 Additional Signalling

Though window-based flow-control mechanisms have been proposed for TFRC [57], they have not been included in the standard. Thus, unlike Freeze-TCP [10], it is not possible to freeze a DCCP/TFRC sender by simply reporting a specific value in a feedback message. Additionally, it is desirable to be able to locally suspend the sender. To fully support freezing on both sides, it is necessary to introduce new signalling options, to be carried in the DCCP or TFRC packet headers. Our proposal does not, however, change or extend the format of these headers; the options will be gracefully ignored by standard implementations.

In a generic typical case, a mobile node, both sending and receiving over the same DCCP connection, will detect or be informed that it is about to lose its current connectivity. In this proposal, packets with an `OPT_FREEZE` DCCP-level option will be sent to the remote peer to suspend its sender operation, then the local sender will be frozen. As the packets carrying the option have to make it to the remote sender, the transport layer has to be informed about upcoming disconnection a short while before it is to happen. A minimum of one RTT is usually considered a reasonable value [10]. This delay is enough for signalling packets to arrive on time (1/2 RTT later) to prevent the transmission of messages which would have arrived after the disconnection (another 1/2 RTT later).

When connectivity becomes available again, the mobile node can restart its traffic and instruct its peer to act similarly by sending packets with an `OPT_UNFREEZE` option.

Additional TFRC-level options are used to support further signalling during the unfreezing phases. The sender uses `OPT_PROBING` and `OPT_RESTORING` to indicate its current state, while the receiver sends an `OPT_UNFROZEN` to signal that it is ready for the Probing phase.

As DCCP is an unreliable protocol, option-carrying packets can be silently lost. Extra care must be taken to ensure both peers are synchronised. This can be done by exchanging options in a redundant manner. The naive approach of adding those to every outgoing packet is chosen here. Depending on the application, this however risks consuming too much capacity and reduction of the option frequency could be considered. As for the `OPT_FREEZE` and `OPT_UNFREEZE` options, an implementation should take care of repeating them on several packets (three in our implementations).

4.4 Frozen Phase

When instructed to freeze, either locally or by the remote peer, the sender enters the Frozen state. In this state, all data transmission ceases. This ensures that no packet will be lost. It in turn guarantees that the loss event rate calculated by the receiver will be kept unmodified. The receiver does not need any specific state for this phase.

The disconnection may however not happen right after freezing, and additional feedback from the receiver may arrive at the sender. Parameters such as the RTT R, or the receiver rate X_{recv} risk being updated. Thus, while in the Frozen state, the sender ignores all feedback messages. When entering this state, it also saves the value of X_{recv} as it will be locally modified on every expiration of the `nofeedback` timer.

To efficiently address longer disconnection periods, and provide some sort of DTN support, it is advisable to additionally increase the connection timeout. Indeed, disconnections longer than 8 minutes may result in the frozen socket being prematurely closed.[5]

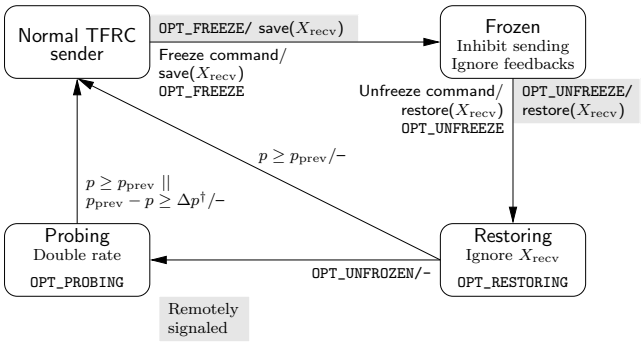
4.5 Restoring Phase

After receiving a local unfreeze instruction or the `OPT_UNFREEZE` option, the sender enters the Restoring state. It first restores X_{recv}. The send timer is then reset to resume packet transmission. As the parameters are the same as before the disconnection, the sending rate will be restored to its previous value.

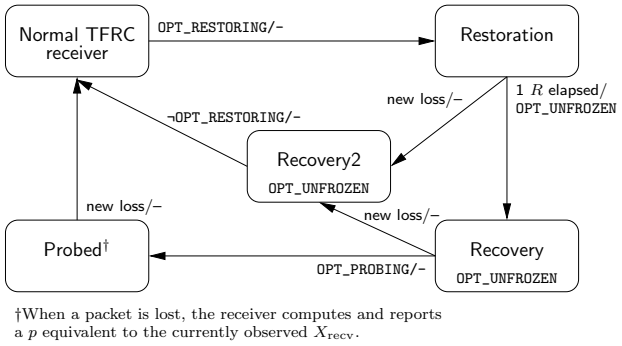
At the same time, it is no longer necessary to completely ignore feedback from the receiver. It is however needed to keep ignoring the X_{recv} reports. Indeed, the receiver rate is measured over at least one RTT. The first feedback packets are likely to cover part of the disconnected period resulting in an incorrectly low value for X_{recv}. Using such value may create instabilities in the sending rate as it is bound by $2X_{\text{recv}}$ as per (2).

When in the Restoring state, the sender adds an `OPT_RESTORING` option to all its outgoing packets to put the receiver into the Restoration state. The Restoring phase ends when the loss event rate increases or an `OPT_UNFROZEN` option is received. This option is added by the receiver after a complete RTT has elapsed, thus signalling that it is no longer necessary to ignore the value of X_{recv} as it will now correctly reflect the receiver rate.

[5] According to [5], a socket in the *Respond* state waits a maximum of four Maximum (TCP) Segment Life for packets before resetting the connection. The Linux 2.6 implementation generalises this idle timeout to the entire lifespan of the socket.



(a) Freeze-DCCP/TFRC sender



(b) Freeze-DCCP/TFRC receiver

Figure 7: Additional states and option exchanges to support Freeze-DCCP/TFRC (transitions are labelled as Condition/Action). The sender (a) can be instructed to freeze or unfreeze either locally or by the remote peer. The receiver (b) does not have to enter a Frozen state, but must perform some specific tasks during the Restoration and Probed phases. Options can signal both/either the remote sender and/or receiver.

4.6 Probing Phase

Standard TFRC quickly reacts to a reduction in the available capacity by responding promptly to an increase in the loss event rate. The `conservative_` mode outlined by [43] further increases this response. Conversely, after idle periods, it is proposed in [45] to increase the sending rate back to the previously supported maximum at an increased pace by quadrupling the rate every RTT.[6] There is however no mechanism to quickly adapt to *better* network conditions. In the Probing state, our sender checks for such improvement in the new network. This is done only if no loss has occurred during the Restoring phase. The sender uses the `OPT_PROBING` option to inform the receiver of its new state. Upon receiving this option, the receiver enters the Probed state.

This phase is similar to a slow-start. Every RTT, the sending rate is doubled. When a loss is detected while it is in the Probed state, the receiver reinitialises its loss history to match the last measured rate. It first computes a packet loss rate p equivalent to the observed receiver rate X_{recv}. It then reinitialises a complete history of n loss intervals of the calculated size.

As p is completely recomputed by the receiver on the first loss, it can be larger, lower or even equal to its previous value. The exit criterion for the probing phase is therefore based on the expected evolution of the reported loss event rate, as derived as (8) in Section 3.3. In a loss-less period, p will never increase. With a growing loss interval, it will however keep decreasing slightly. The sender should thus exit the Probing state if $\Delta p \notin]\Delta p_{\min}(X_{\text{Bps}} \cdot R, p_{\text{prev}}); 0[$, following (9). Missing `OPT_PROBING` options on new packets then takes the receiver out of the Probed state.

It may happen that the sender-recomputed p lies in the acceptable range of variation. In this case, the sender cannot detect that the Probing phase should be ended. Some more losses will however be generated during the next RTT. These losses will prevent p from changing during the next report, thus properly ending the Probing phase as per the previous criterion.

[6] This draft proposal considers application-limited rates rather than disconnections; it also does not restore the rate at once as our Restoring phase does.

Table 5: Characteristics of the wireless networks used for evaluation purposes.

Technology	D/L capacity [bps]	Avg. RTT [s]
UMTS	384 k	250 m [59]
802.11b/g	11 M/54 M	20 m [60]
802.16	9.5 M [61]	80 m [62]

5 Performance Evaluation

This section presents an evaluation of the enhancements proposed in the previous section. It first compares, in *ns*-2 simulations, the behaviour of Freeze-DCCP/TFRC with that of the unmodified version. It then demonstrates that the proposed mechanism still retains a satisfying level of fairness to TCP flows. Finally, a real experiment, based on a Linux implementation, shows that our proposal is well suited to improve the QoE of a live video stream experiencing multiple handovers between heterogeneous technologies.

5.1 Realistic Handover Simulations

Simulations were run with *ns*-2.33. Additional modifications have been made to the TFRC sender of the DCCP module [50] to implement the clarified loss average calculation of [8]. The `DCCP/TFRC/Freeze` agent has been implemented by deriving the DCCP/TFRC C++ class to add the freezing mechanisms described above.

As the impact of disconnections is only relevant to our study at the transport layer, the underlying wireless technologies are not simulated as such. Rather, simple wired topologies are used, as suggested by [58]. All wireless links are modelled as duplex links, even the wireless ones. This may not be correct for bi-directional data scenarios. Such scenarios, however, are not considered here. The characteristics of the various technologies used in these simulation are taken from the respective standards and measurement-based literature, and summarised in Table 5.

A router is placed between the sender and the receiver. Such a topology allows to transparently disconnect the link between the router and the receiver without preventing the sender from trying to transmit packets during the discon-

Table 6: Simulated MIPv6 handovers performance impact for DCCP/TFRC and Freeze-DCCP/TFRC (grey).

from \ to	UMTS	802.16	802.11 b	802.11 g
Packet losses (DCCP/TFRC only)				
UMTS	253.3	269.8	273.6	275.4
802.16	1732.3	1734.6	1734.6	1734.6
802.11b	856	855.5	855.3	855.3
802.11g	2470.9	2470.4	2470.2	2470.1
Wasted capacity [Number of 500 B packets]				
UMTS	50.5	54018.05	2190.45	92156.1
—	13.4	3607.9	9342.75	89328.6
802.16	12.45	1827.95	603.05	4185.75
—	5	591.15	150.9	1520.35
802.11b	150.45	28314	2101.75	57970.65
—	0	15278	47.45	1045.05
802.11g	42.5	2104.3	943.4	4313
—	0	7172.75	46.5	188.45

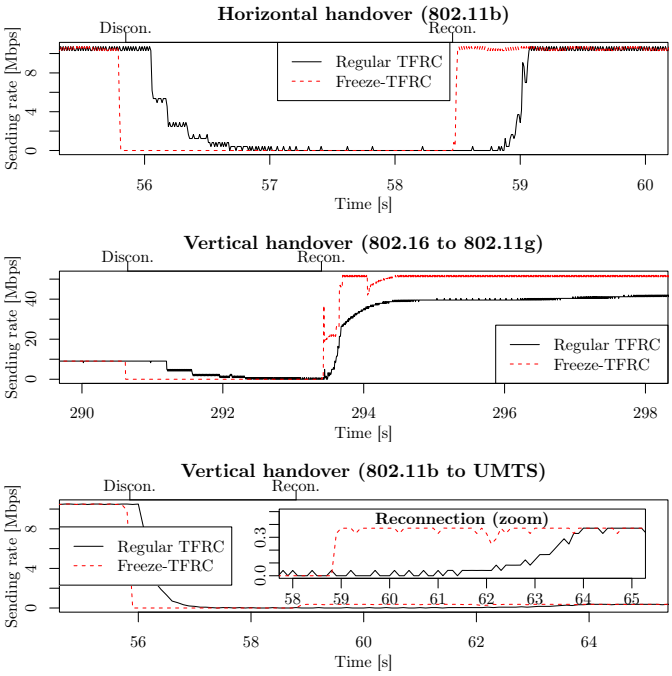


Figure 8: Comparison of the rate of DCCP/TFRC and the Freeze-enabled version in typical examples of MIPv6 horizontal or vertical handovers.

nections. `DropTail` queues with the default buffer size (50 packets) are used.

Disconnections are simulated by manipulating the routing model of *ns*-2 with the `$ns_ rtmodel-at` function. The time of the hand-offs is chosen, once the system is in a stationary state, from a uniformly distributed variable over a time period of four RTTs. The generic behaviour of both standard TFRC and our variant is thus captured. Link characteristics are modified using the `$ns_ bandwidth` and `$ns_ delay` commands. The simulations were ended after the rates had settled on the new network. The result have been averaged over 20 runs.

The Freeze-DCCP agent is instructed to suspend its connection locally (*i.e.*, not using the `OPT_FREEZE` and `OPT_UNFREEZE` options). The `freeze` command is given one RTT before the disconnection is scheduled to happen (as suggested by [10]). The `unfreeze` instruction is given 0.1 ms after the network link is reconnected.

The number of losses upon reconnection, as well as the wasted capacity, are shown on Table 6. As Freeze-DCCP/TFRC did not lose any packet, this information is omitted. The wasted capacity has been estimated by comparing TFRC's actual rate X to what is achievable in the steady state (Table 3), then converted in number of 500 B packets. Figure 8 illustrates these figures by comparing how both regular DCCP and the Freeze-enabled version perform in key example scenarios.

Two cases stand out in Table 6 where the average amount of wasted capacity is larger with our proposal than with standard TFRC. These correspond to pathological cases for our *ns*-2 implementation where either the Recovery or the Probing phase experiences losses too early. The rate reported by the receiver does not cover a full RTT (802.11 g to 802.16), or the probing phase terminates before having discovered the network capacity (UMTS to 802.11 b). Future work should address this type of situation by using other or additional metrics, rather than just losses, to drive the Restoration/Probing phases.

Overall, Freeze-DCCP/TFRC quickly restores an acceptable rate for the application, and greatly reduces the overall under-usage of the available capacity upon reconnection. It also prevents traffic after the hand-off to unnecessarily use the rest of the network.

5.2 Fairness to TCP Flows

TFRC was designed to quickly respond to reductions of the capacity. The restoring and probing features of Freeze-DCCP/TFRC, however, aggressively test and use the network. TCP-friendliness is however a key property of TFRC, which it is desirable to retain. It is therefore important to check that our additions do not make the protocol too greedy. Though [20] argues that usage fairness should not be based on rate comparisons, such an approach is still commonly accepted, and we use it here.

The criterion to evaluate TCP-fairness is the ratio of the average capacity occupation of Freeze-DCCP/TFRC to that of concurrent TCP flows. The samples, taken after the reconnection, have been averaged over 100 s, discarding the initial rate settlement period.

Table 7 compares the average fairness of a (Freeze-)DCCP/TFRC flow to a concurrent TCP stream, as observed after the reconnection for the studied handover scenarios. The proposed improvement appears to be reasonably fair to TCP flows in various scenarios including vertical handovers to technologies with higher or lower capacities. In some cases it is even too fair, not competing aggressively enough for the network. A similar behaviour is however also observed for the regular DCCP/TFRC, and is not a result of our changes.

Table 7: Fairness comparison of both standard TFRC and the Freeze-enabled proposal to TCP after a handover (taking network parameters from Table 5). Values in the range [0.5; 2] are considered "reasonably fair" [8].

from \ to	UMTS	802.16	802.11 b	802.11 g
Standard DCCP/TFRC				
UMTS	1.3	0.7	0.4	0.3
802.16	1.1	1	0.9	0.8
802.11b	1.2	1	1	0.8
802.11g	1.3	1.1	1	1.1
Freeze-DCCP/TFRC				
UMTS	0.6	0.3	0.2	0.1
802.16	1.6	1.3	1.1	0.9
802.11b	1.3	1	0.9	0.7
802.11g	1.5	1.2	1	1.1

5.3 QoE of Mobile Video Streaming

To explore the actual performance improvements of our proposal, we used an OMF testbed [14] to emulate vertical handovers and observe the impact on the quality of a video stream. An implementation of Freeze-DCCP/TFRC was developed in the Linux kernel[7] and used here.

The scenario is depicted in Figure 9.[8] A user, initially at home (t_0), receives a video stream on their mobile terminal connected to their home Wi-Fi network connected to their DSL link (1 Mbps). They decide to get a coffee from the corner shop. On the way there, the mobile terminal loses its connectivity to the home network, and hands off to a shared 3G network (t_1; 500 kbps). The coffee shop has a public wireless network (*e.g.*, a shared Internet access, offering 700 kbps of capacity), to which the device connects when it arrives in range (t_2). With their coffee in hand, the user then heads back home, losing connectivity to the public Wi-Fi network and performing a new handover to 3G at t_3 before finally reconnecting to their home network at t_4. Throughout the streaming period, the device thus goes through several handovers between various wireless networks with different capacities and delays.

In this experiment, a video file, encoded and packetised using the H.264 codec with a 1 Mbps bit-rate, is sent using a specially patched version of Iperf [63][9] to a custom receiver application. Both endpoints provide information about the sending or receiving rates. The custom receiver identifies lost packets and computes a moving average of the PSNR over 24 frames ($\simeq$ 1 s), using the `compare` tool of the ImageMagick package [64].[10] Both sender and receiver have been instrumented with the OML toolkit [65]. This allows

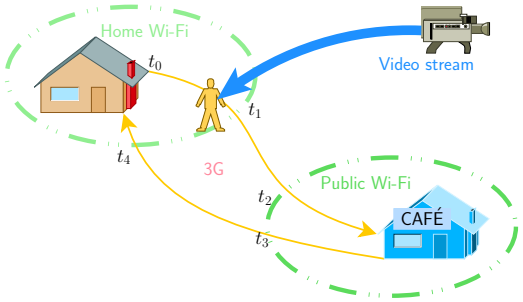


Figure 9: Scenario for the evaluation of Freeze-DCCP/TFRC improvements on application quality. A user viewing a video stream on their mobile device goes out for a coffee then comes back home. In the process, networks with heterogeneous characteristics are visited.

them to report readings of these metrics in real time for analysis or display.

Our OMF testbed currently only supports 802.11-based wireless networks. However, only a limited set of parameters of the underlying network is relevant at the transport layer. Therefore, as for the previous section, we follow the suggestion of [58] and emulate the conditions of wireless technologies in different networks by shaping the available capacity using Linux' traffic control tools [66], and regulating forwarding delays using NetEm [67].

The results, averaged over 8 runs of our scenario, comparing the PSNR of the video stream when using Freeze-DCCP/TFRC to that with the standard version, are shown in Figure 10. They show that the use of Freeze-DCCP/TFRC results in a reduced but stable QoE when on those networks which cannot support 1 Mbps streams, as our proposal is able to adapt much faster and use close to the full available path capacity to carry application data. In comparison, the PSNR of a video stream supported by the regular DCCP/TFRC reduces to a minimum (a PSNR of 7 dB is that of a purely random image), and takes a long time (up to the complete visit duration of a network) to restore to a better level. In addition, Freeze-DCCP/TFRC does not suffer from the oscillations which appear for the standard version when visiting the Café's Wi-Fi network. We hypothesise that it is due to the probing mechanism finding the capacity of the new network more accurately, and its estimates not being biased by measurements from the previous network. It might be interesting to instrument the kernel code of DCCP (*e.g.*, TFRC's loss history) in order to investigate this further.

6 Conclusion and Future Work

In this article, we have first identified the issues that TFRC faces in mobility situations. We have numerically modelled the losses and subsequent under-usage of the available capacity that it experiences in those cases. This model allowed us to evaluate the performance improvements that could be expected from a system with a better awareness and handling of disconnections. We thus proposed Freeze-DCCP/TFRC, an extension of the TFRC congestion control mechanism used by DCCP, to approach these possible performance gains. This proposal is aimed at uses of DCCP in situations where network connectivity may periodically

[7] We used the Net:DCCP tree (http://eden-feed.erg.abdn.ac.uk/cgi-bin/gitweb.cgi?p=dccp_exp.git;a=summary),forked off of vanilla version 2.6.34-rc5, as a starting point. A Git branch containing these modifications is available at http://github.com/shtrom/linux-2.6/tree/freezedccp.

[8] The OMF scripts are available at http://omf.mytestbed.net/projects/omf-case-studies/wiki/FreezeDccpQoE.

[9] This version, supporting both DCCP and OML, is available from http://www.nicta.com.au/people/mehanio/freezedccp#iperf-dccp-oml.

[10] This tool computes the PSNR as the log of the mean square error between an image and its reference.

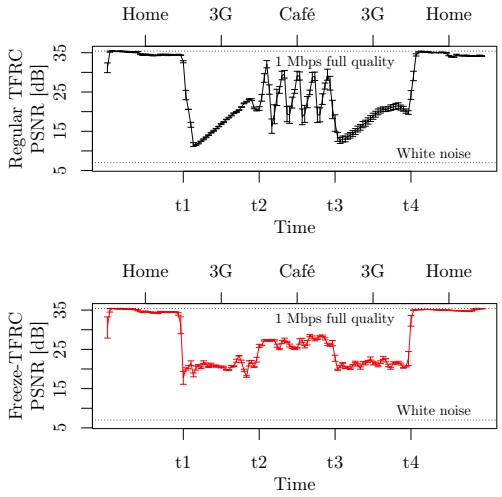


Figure 10: PSNR comparison for a video stream using TFRC or Freeze-TFRC in mobility situations. Averages over 8 runs; error bars indicate the standard error.

not be available for varying periods of time, and the access networks' characteristics may widely vary between disconnections.

Freeze-DCCP/TFRC was both implemented in *ns*-2 and Linux. Simulation results have shown that it is possible to prevent handover-induced losses, to restore the rate faster when reconnecting to a link with lower or similar capacities and to adapt more quickly to higher capacities. Additionally, we confirmed that our proposal maintains the important TFRC's property of being fair to concurrent TCP flows. We have also experimentally shown that our proposal can significantly improve the performance quality of streaming applications when disconnections are predictable, for example, for IP mobility or. Though the proposed modifications were designed with real-time applications over DCCP in mind, we believe they are versatile enough in terms of rate adaptation and packet loss avoidance to also benefit other types of traffic.

Additional work may however be needed for the proposed extension to be used in real deployments. First, it would be desirable to decouple the freezing mechanism, which caters for hand-off-induced disconnections, and the probing phase, which deals with heterogeneous paths. Indeed, only the latter is needed for make-before-break handovers. Second, the current probing mechanism assumes packets are lost only when the capacity of the new network is reached. Other methods of detecting that the current rate matches the available path capacity should be explored. Both in-band solutions, such as MBTFRC, and out-of-band ones could be considered.

References

[1] D. B. Johnson, C. E. Perkins, and J. Arkko, "Mobility support in IPv6," RFC 6275, Jul. 2011. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6275.txt.

[2] V. Tsaoussidis and I. Matta, "Open issues on TCP for mobile computing," *Wireless Communications and Mobile Computing*, vol. 2, no. 1, pp. 3–20, Feb. 2002, ISSN: 1530-8677. DOI: 10.1002/wcm.30. [Online]. Available: http://dx.doi.org/10.1002/wcm.30.

[3] S. Floyd and K. Fall, "Promoting the use of end-to-end congestion control in the internet," *IEEE/ACM Transactions on Networking*, vol. 7, no. 4, pp. 458–472, Aug. 1999, ISSN: 1063-6692. DOI: 10.1109/90.793002. [Online]. Available: http://icir.org/floyd/papers/collapse.may99.pdf.

[4] E. Kohler, M. Handley, and S. Floyd, "Designing DCCP: congestion control without reliability," *SIGCOMM Computer Communication Review*, vol. 36, no. 4, pp. 27–38, Oct. 2006, ISSN: 0146-4833. DOI: 10.1145/1151659.1159918. [Online]. Available: http://www.cs.ucla.edu/~kohler/pubs/kohler06designing.pdf.

[5] ——, "Datagram congestion control protocol (DCCP)," RFC 4340, Mar. 2006. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4340.txt.

[6] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," *SIGCOMM Computer Communication Review*, vol. 30, no. 4, pp. 43–56, Oct. 2000, ISSN: 0146-4833. DOI: 10.1145/347057.347397. [Online]. Available: http://www.icir.org/tfrc/tcp-friendly.pdf.

[7] J. Widmer, "Equation-based congestion control for unicast and multicast data streams," PhD thesis, May 2003. [Online]. Available: http://icapeople.epfl.ch/widmer/files/Widmer2003a.pdf.

[8] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "TCP friendly rate control (TFRC): protocol specification," RFC 5348, Sep. 2008. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5348.txt.

[9] A. Gurtov and J. Korhonen, "Effect of vertical handovers on performance of TCP-friendly rate control," *SIGMOBILE Mobile Computing and Communications Review*, vol. 8, no. 3, pp. 73+, Jul. 2004, ISSN: 1559-1662. DOI: 10.1145/1031483.1031494. [Online]. Available: http://www.cs.helsinki.fi/u/gurtov/papers/vho.pdf.

[10] T. Goff, J. Moronski, D. S. Phatak, and V. Gupta, "Freeze-TCP: a true end-to-end TCP enhancement mechanism for mobile environments," in *INFOCOM 2000*, vol. 3, Mar. 2000, pp. 1537–1545, ISBN: 0-7803-5880-5. DOI: 10.1109/INFCOM.2000.832552. [Online]. Available: http://www.cs.umbc.edu/~phatak/publications/ftcp.pdf.

[11] N. Montavont and T. Noël, "Stronger interaction between link layer and network layer for an optimized mobility management in heterogeneous IPv6 networks," *Pervasive and Mobile Computing*, vol. 2, no. 3, pp. 233–261, Sep. 2006, ISSN: 1574-1192. DOI: 10.1016/j.pmcj.2006.02.001. [Online]. Available: http://dx.doi.org/10.1016/j.pmcj.2006.02.001.

[12] K. Kilkki, "Quality of experience in communications ecosystem," *Journal of Universal Computer Science*, vol. 14, no. 5, pp. 615–624, Mar. 2008. [Online]. Available: http://www.jucs.org/jucs_14_5/quality_of_experience_in/jucs_14_05_0615_0624_kilkki.pdf.

[13] ANSI T1.TR.74-2001, "Objective video quality measurement using a peak-signal-to-noise-ratio (PSNR) full reference technique," Tech. Rep. T1.TR.74-2001, 2001. [Online]. Available: http://webstore.ansi.org/RecordDetail.aspx?sku=T1.TR.74-2001.

[14] T. Rakotoarivelo, M. Ott, G. Jourjon, and I. Seskar, "OMF: a control and management framework for networking nestbeds," *SIGOPS Operating Systems Review*, vol. 43, no. 4, pp. 54–59, Jan. 2010, ISSN: 0163-5980. DOI: 10.1145/1713254.1713267. [Online]. Available: http://www.nicta.com.au/research/research_publications/show?id=2155.

[15] F. Nazir and A. Seneviratne, "Towards mobility enabled protocol stack for future wireless network," *Ubiquitous Computing and Communication Journal*, vol. 2, no. 4, Aug. 2007. [Online]. Available: http://www.ubicc.org/abstract.aspx?id=63.

[16] W. M. Eddy, "At what layer does mobility belong?" *IEEE Communications Magazine*, vol. 42, no. 10, pp. 155–159, Oct. 2004, ISSN: 0163-6804. DOI: 10.1109/MCOM.2004.1341274. [Online]. Available: http://dx.doi.org/10.1109/MCOM.2004.1341274.

[17] M. Allman, V. Paxson, and E. Blanton, "TCP congestion control," RFC 5681, Sep. 2009. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5681.txt.

[18] S. Floyd, T. Henderson, and A. Gurtov, "The NewReno modification to TCP's fast recovery algorithm," RFC 3782, Apr. 2004. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3782.txt.

[19] S. Floyd, "Metrics for the evaluation of congestion control mechanisms," RFC 5166, Mar. 2008. [Online]. Available: http://www.rfc-editor.org/rfc/rfc5166.txt.

[20] B. Briscoe, "Flow rate fairness: dismantling a religion," *SIGCOMM Computer Communication Review*, vol. 37, no. 2, Apr. 2007, ISSN: 0146-4833. DOI: 10.1145/1232919.1232926. [Online]. Available: http://bobbriscoe.net/projects/2020comms/refb/fair_ccr.pdf.

[21] R. R. Stewart, "Stream control transmission protocol," RFC 4960, Sep. 2007. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4960.txt.

[22] Y. Han and F. Teraoka, "SCTPfx: a fast failover mechanism based on cross-layer architecture in SCTP multihoming," in *AINTEC 2008*, Nov. 2008, pp. 113–122, ISBN: 978-1-60558-127-9. DOI: 10.1145/1503370.1503399. [Online]. Available: http://dx.doi.org/10.1145/1503370.1503399.

[23] Y. Han and F. Terakoa, "SCTPmx: an SCTP fast handover mechanism using a single interface based on a cross-layer architecture," *IEICE Transactions on Communications*, vol. E.92B, no. 9, pp. 2864–2873, Sep. 2009, ISSN: 1745-1345. [Online]. Available: http://search.ieice.org/bin/summary.php?id=e92-b_9_2864&category=B&year=2009&lang=E&abst=.

[24] S. Floyd and E. Kohler, "Profile for datagram congestion control protocol (DCCP) congestion control ID 2: TCP-like congestion control," RFC 4341, Mar. 2006. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4341.txt.

[25] S. Floyd, E. Kohler, and J. Padhye, "Profile for datagram congestion control protocol (DCCP) congestion control ID 3: TCP-friendly rate control (TFRC)," RFC 4342, Mar. 2006. [Online]. Available: http://www.rfc-editor.org/rfc/rfc4342.txt.

[26] S. Floyd and E. Kohler, "Profile for datagram congestion control protocol (DCCP) congestion ID 4: TCP-friendly rate control for small packets (TFRC-SP)," RFC 5622, Aug. 2009. [Online]. Available: http://www.rfc-editor.org/rfc/draft-ietf-dccp-ccid4.txt.

[27] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, "Modeling TCP throughput: a simple model and its empirical validation," *SIGCOMM Computer Communication Review*, vol. 28, no. 4, pp. 303–314, Oct. 1998, ISSN: 0146-4833. DOI: 10.1145/285243.285291. [Online]. Available: http://dx.doi.org/10.1145/285243.285291.

[28] K. Chen, K. Nahrstedt, and N. H. Vaidya, "The utility of explicit rate-based flow control in mobile ad hoc networks," in *WCNC 2004*, vol. 3, Mar. 2004, pp. 1921–1926, ISBN: 0-7803-8344-3. DOI: 10.1109/WCNC.2004.1311847. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1311847.

[29] S. Mascolo, C. Casetti, M. Gerla, M. Y. Sanadidi, and R. Wang, "TCP Westwood: bandwidth estimation for enhanced transport over wireless links," in *MobiCom 2001*, Jul. 2001, pp. 287–297, ISBN: 1-58113-422-3. DOI: 10.1145/381677.381704. [Online]. Available: http://www.cs.ucla.edu/NRL/hpi/tcpw/tcpw_papers/2001-mobicom-0.pdf.

[30] L. A. Grieco and S. Mascolo, "Performance evaluation and comparison of Westwood+, New Reno, and Vegas TCP congestion control," *SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 25–38, Apr. 2004, ISSN: 0146-4833. DOI: 10.1145/997150.997155. [Online]. Available: http://dx.doi.org/10.1145/997150.997155.

[31] H. Balakrishnan, V. N. Padmanabhan, S. Seshan, and R. H. Katz, "A comparison of mechanisms for improving TCP performance over wireless links," *IEEE/ACM Transactions on Networking*, vol. 5, no. 6, pp. 756–769, Dec. 1997, ISSN: 1063-6692. DOI: 10.1109/90.650137. [Online]. Available: http://www.stanford.edu/class/cs244e/papers/balakrishnan_ton.pdf.

[32] G. Xylomenos, G. C. Polyzo, P. Mähönen, and M. Saaranen, "TCP performance issues over wireless links," *IEEE Communications Magazine*, vol. 39, no. 4, pp. 52–58, Apr. 2001, ISSN: 0163-6804. DOI: 10.1109/35.917504. [Online]. Available: http://dx.doi.org/10.1109/35.917504.

[33] S. Pilosof, R. Ramjee, D. Raz, Y. Shavitt, and P. Sinha, "Understanding TCP fairness over wireless LAN," in *INFOCOM 2003*, vol. 2, Apr. 2003, pp. 863–872. DOI: 10.1109/INFCOM.2003.1208924. [Online]. Available: http://dx.doi.org/10.1109/INFCOM.2003.1208924.

[34] K. Benekos, N. Pogkas, G. Kalivas, G. Papadopoulos, and A. Tzes, "TCP performance measurements in IEEE 802.11b-based wireless LANs," in *MELECON 2004. 12th IEEE Mediterranean Electrotechnical Electrotechnical Conference*, vol. 2, May 2004, pp. 575–578, ISBN: 0-7803-8271-4. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1346995.

[35] M. Franceschinis, M. Mellia, M. Meo, and M. Munafò, "Measuring TCP over WiFi: a real case," in *WiNMee 2005*, Apr. 2005. [Online]. Available: http://www.winmee.org/2005/papers/WiNMee_Franceschinis.pdf.

[36] A. Baig, L. Libman, and M. Hassan, "Performance enhancement of on-board communication networks using outage prediction," *IEEE Journal on Selected Areas in Communications*, vol. 24, no. 9, pp. 1692–1701, Sep. 2006, ISSN: 0733-8716. DOI: 10.1109/JSAC.2006.875108. [Online]. Available: http://dx.doi.org/10.1109/JSAC.2006.875108.

[37] M. Park, J. Lee, J. Koo, and H. Choo, "Freeze TCPv2: an enhancement of Freeze TCP for efficient handoff in heterogeneous networks," in *Human Interface and the Management of Information. Information and Interaction*, ser. Lecture Notes in Computer Science, vol. 5618, 2009, ch. 49, pp. 448–457, ISBN: 978-3-642-02558-7. DOI: 10.1007/978-3-642-02559-4_49. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-02559-4_49.

[38] L. Zhang, P. Sénac, E. Lochin, and M. Diaz, "Cross-layer based congestion control for WLANs," in *QShine 2008*, Jun. 2008, pp. 1–7, ISBN: 978-963-9799-26-4. [Online]. Available: http://portal.acm.org/citation.cfm?id=1535619.

[39] C. S. Shieh, I. C. Lin, and W. K. Lai, "Improvement of SCTP performance in vertical handover," in *ISDA 2008*, vol. 3, 2008, pp. 494–498, ISBN: 978-0-7695-3382-7. DOI: 10.1109/ISDA.2008.347. [Online]. Available: http://dx.doi.org/10.1109/ISDA.2008.347.

[40] Y. Lin, S. Cheng, W. Wang, and Y. Jin, "Measurement-based TFRC: improving TFRC in heterogeneous mobile networks," *IEEE Transactions on Wireless Communications*, vol. 5, no. 8, pp. 1971–1975, Aug. 2006, ISSN: 1536-1276. DOI: 10.1109/TWC.2006.1687706. [Online]. Available: http://dx.doi.org/10.1109/TWC.2006.1687706.

[41] L.-J. Chen, T. Sun, G. Yang, M. Y. Sanadidi, and M. Gerla, "Monitoring access link capacity using TFRC probe," *Computer Communications*, vol. 29, no. 10, pp. 1605–1613, Jun. 2006, ISSN: 01403664. DOI: 10.1016/j.comcom.2005.07.010. [Online]. Available: http://www.cs.ucla.edu/NRL/CapProbe/.

[42] C. Dovrolis, P. Ramanathan, and D. Moore, "What do packet dispersion techniques measure?" In *INFOCOM 2001*, Apr. 2001, pp. 905–914. DOI: 10.1109/INFCOM.2001.916282. [Online]. Available: http://minds.wisconsin.edu/bitstream/handle/1793/9304/file_1.pdf.

[43] D. Bansal, H. Balakrishnan, S. Floyd, and S. Shenker, "Dynamic behavior of slowly-responsive congestion control algorithms," in *SIGCOMM 2001*, Aug. 2001, pp. 263–274, ISBN: 1-58113-411-8. DOI: 10.1145/383059.383080. [Online]. Available: http://nms.lcs.mit.edu/papers/slowcc-sigcomm01.html.

[44] D. Li, K. Sleurs, E. Van Lil, and A. Van de Capelle, "Improving TFRC performance against bandwidth change during handovers," in *WiCom 2008*, Oct. 2008, pp. 1–4, ISBN: 978-1-4244-2107-7. DOI: 10.1109/WiCom.2008.1072. [Online]. Available: http://lirias.kuleuven.be/bitstream/123456789/199721/1/PID667147.pdf.

[45] E. Kohler, S. Floyd, and A. Sathiaseelan, "Faster restart for TCP friendly rate control (TFRC)," Internet-Draft draft-ietf-dccp-tfrc-faster-restart-06.txt, Jul. 2008. [Online]. Available: http://www.rfc-editor.org/internet-drafts/draft-ietf-dccp-tfrc-faster-restart-06.txt.

[46] G. Jourjon, E. Lochin, and L. Dairaine, "Optimization of TFRC loss history initialization," *IEEE Communications Letters*, vol. 11, no. 3, pp. 276–278, Mar. 2007, ISSN: 1089-7798. DOI: 10.1109/LCOMM.2007.061707. [Online]. Available: http://dx.doi.org/10.1109/LCOMM.2007.061707.

[47] *The ns manual (formerly ns notes and documentation)*, Jan. 2009. [Online]. Available: http://www.isi.edu/nsnam/ns/doc/index.html.

[48] D. Johnson, C. E. Perkins, and J. Arkko, "Mobility support in IPv6," RFC 3775, Jun. 2004. [Online]. Available: http://www.rfc-editor.org/rfc/rfc3775.txt.

[49] T. Ernst, *MobiWan: a ns-2.1b6 simulation platform for mobile IPv6 in wide area networks*, May 2001. [Online]. Available: http://www.inrialpes.fr/planete/mobiwan/.

[50] N.-E. Mattsson, "A DCCP module for ns-2," Master's thesis, May 2004. [Online]. Available: http://epubl.luth.se/1402-1617/2004/175/index-en.html.

[51] T. J. Ypma, "Historical development of the Newton–Raphson method," *SIAM Review*, vol. 37, no. 4, pp. 531–551, Dec. 1995, ISSN: 0036-1445. DOI: 10.1137/1037125. [Online]. Available: http://dx.doi.org/10.1137/1037125.

[52] R. Koodli and C. E. Perkins, "Fast handovers and context transfers in mobile networks," *SIGCOMM Computer Communication Review*, vol. 31, no. 5, pp. 37–47, Oct. 2001, ISSN: 0146-4833. DOI: 10.1145/1037107.1037113. [Online]. Available: http://www.ieee802.org/21/archived_docs/2003-09_incoming/ccr-200109-koodli.pdf.

[53] J. S. Lee, S. J. Koh, and S. H. Kim, "Analysis of handoff delay for mobile IPv6," in *VTC2004-Fall, 60th IEEE Vehicular Technology Conference*, vol. 4, Sep. 2004, 2967–2969 Vol. 4, ISBN: 0-7803-8521-7. DOI: 10.1109/VETECF.2004.1400604. [Online]. Available: http://dx.doi.org/10.1109/VETECF.2004.1400604.

[54] H. Balakrishnan, H. S. Rahul, and S. Seshan, "An integrated congestion management architecture for Internet hosts," *SIGCOMM Computer Communication Review*, vol. 29, no. 4, pp. 175–187, Oct. 1999, ISSN: 0146-4833. DOI: 10.1145/316194.316220. [Online]. Available: http://dx.doi.org/10.1145/316194.316220.

[55] O. Mehani, R. Boreli, M. Maher, and T. Ernst, "User- and application-centric multihomed flow management," in *LCN 2011*, Oct. 2011, pp. 26–34, ISBN: 978-1-61284-928-7. [Online]. Available: http://www.nicta.com.au/pub?id=4578.

[56] E. Piri and K. Pentikousis, "IEEE 802.21," *The Internet Protocol Journal*, vol. 12, no. 2, pp. 7–27, Jun. 2009. [Online]. Available: http://www.cisco.com/web/about/ac123/ac147/archived_issues/ipj_12-2/122_ieee.html.

[57] E. Lochin, G. Jourjon, S. Ardon, and P. Senac, "Promoting the use of reliable rate-based transport protocols: the Chameleon protocol," *International Journal of Internet Protocol Technology*, vol. 5, no. 4, pp. 175–189, 2010, ISSN: 1743-8217. DOI: 10.1504/IJIPT.2010.039229. [Online]. Available: http://www.nicta.com.au/pub?doc=3015.

[58] A. Gurtov and S. Floyd, "Modeling wireless links for transport protocols," *SIGCOMM Computer Communication Review*, vol. 34, no. 2, pp. 85–96, Apr. 2004, ISSN: 0146-4833. DOI: 10.1145/997150.997159. [Online]. Available: http://dx.doi.org/10.1145/997150.997159.

[59] F. Vacirca, F. Ricciato, and R. Pilz, "Large-scale RTT measurements from an operational UMTS/GPRS network," in *WICON 2005*, Jul. 2005, pp. 190–197, ISBN: 0-7695-2382-X. DOI: 10.1109/WICON.2005.19. [Online]. Available: http://userver.ftw.at/~ricciato/darwin/wicon05-ricciato-metawin.pdf.

[60] T. Karapantelakis and G. Iacovidis, "Experimenting with real time applications in an IEEE 802.11b ad hoc network," in *LCN 2005*, vol. 0, Nov. 2005, pp. 554–559. DOI: 10.1109/LCN.2005.67. [Online]. Available: http://dx.doi.org/10.1109/LCN.2005.67.

[61] O. Grøndalen, P. Grønsund, T. Breivik, and P. Engelstad, "Fixed WiMAX field trial measurements and analyses," in *16th IST Mobile and Wireless Communications Summit*, Jul. 2007, pp. 1–5, ISBN: 1-4244-1662-0. DOI: 10.1109/ISTMWC.2007.4299213. [Online]. Available: http://folk.uio.no/paalee/publications/wimax-mobilesummit-2007.pdf.

[62] E. Halepovic, Q. Wu, C. Williamson, and M. Ghaderi, "TCP over WiMAX: a measurement study," in *MASCOTS 2008*, Sep. 2008, pp. 1–10, ISBN: 978-1-4244-2817-5. DOI: 10.1109/MASCOT.2008.4770565. [Online]. Available: http://dx.doi.org/10.1109/MASCOT.2008.4770565.

[63] M. Gates, A. Tirumala, J. Dugan, and K. Gibbs, *Iperf version 2.0.0*, May 2004. [Online]. Available: http://iperf.sf.net.

[64] M. Still, *The Definitive Guide to ImageMagick*. Dec. 2005, ISBN: 978-1590595904. [Online]. Available: http://www.worldcat.org/isbn/1590595904.

[65] O. Mehani, G. Jourjon, T. Rakotoarivelo, and M. Ott, *An instrumentation framework for the critical task of measurement collection in the future Internet*, Under review, 2012.

[66] B. Hubert, T. Graf, G. Maxwell, R. Van Mook, M. Van Oosterhout, P. B. Schroeder, J. Spaans, and P. Larroy, *Linux advanced routing & traffic control HOWTO*, Apr. 2004. [Online]. Available: http://lartc.org/lartc.pdf.

[67] S. Hemminger, "Network emulation with NetEm," in *LCA 2005*, Apr. 2005. [Online]. Available: http://www.linux.org.au/conf/2005/abstract2e37.html?id=163.