Analysis of TFRC in Disconnected Scenarios and Performance Improvements with Freeze-DCCP

Olivier Mehani
National ICT Australia, and University of New South Wales, Australia, and Mines ParisTech, CAOR — Centre de Robotique, Mathématiques et Systèmes, France
olivier.mehani@nicta.com.au

Roksana Boreli
National ICT Australia
Locked Bag 9013
Alexandria NSW 1435, Australia, and University of New South Wales, Australia
roksana.boreli@nicta.com.au

Thierry Ernst
Inria Rocquencourt, Imara project-team, Domaine de Voluceau, BP 105
78153 Le Chesnay Cedex, France
thierry.ernst@inria.fr

ABSTRACT
We present enhancements to the TCP-Friendly Rate Control mechanism (TFRC) which are designed to better cope with the intermittent connectivity available to mobile devices or in Delay Tolerant Networks. Our aim is to prevent losses during disconnected periods and quickly adapt to new network conditions. We propose to suspend the transmission before disconnections occur in a way inspired by Freeze-TCP, then probe, in a new way, the network after reconnecting to enable full use of the newly available bandwidth. We first evaluate the potential performance gains for realistic network parameters. We then describe the proposed additions to TFRC and their implementation within the Datagram Congestion Control Protocol (DCCP) in ns-2. Comparisons of simulation results for example mobility scenarios show that the proposed enhancements enable faster recovery upon reconnection as well as significantly improved adjustment to the newly available network conditions.

Categories and Subject Descriptors
C.2.2 [**Computer-Communication Networks**]: Network Protocols

General Terms
Design, Performance

Keywords
DCCP, TFRC, congestion control, IPv6 mobility, delay tolerant networks

1. INTRODUCTION
Connectivity for mobile devices has become a norm in recent years. The technologies used to establish such connections are manifold (*e.g.* 802.11, 802.16, GPRS or UMTS). Roaming between several of these networks can be handled by various techniques. Among these, IP mobility schemes such as MIPv6 [4] deal with the inherent dynamicity of such scenarios at the network layer.

With the increased connectivity, there is also an increasing use of real time applications like multimedia and IP telephony. Such applications have a need for timely data, however they must not starve other applications of network resources.

UDP, usually used to carry real-time traffic, does not provide congestion control. TCP, on the other hand, competes fairly with other network flows but is not well suited for real-time traffic. Indeed, retransmitted packets may no longer be needed by the time they reach the receiver. The Datagram Congestion Control Protocol [5] has been proposed as a non-reliable but congestion-aware transport protocol. DCCP can make use of the TCP-Friendly Rate Control mechanism [1], which replicates TCP's response to adjust to the network capacity.

Even though network mobility schemes hide most of the complexity of roaming to the upper layers, cross-technology handoffs 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 [6, 3].

In this paper, we propose to analytically model the effects of such handovers on TFRC in order to quantitatively derive the potential for improvement. We then present an end-to-end solution to better cope with such events. This disconnection-tolerant modification of DCCP/TFRC is inspired by the concept of Freeze-TCP [2] but introduces further enhancements, including a sender-directed freezing of the connection, and an improved reconnection phase with faster adaptation to available bandwidth through probing. Information about upcoming handover disconnections and reconnections is assumed to be reliably available.

This paper is structured as follows: Section 2 provides an overview of the related work; Section 3 introduces an analytical study of the behavior of TFRC when disconnections occur, allowing to estimate the possible performance gains; Section 4 presents the proposed TFRC protocol modifications and their implementation into Freeze-DCCP; simulation results are presented in section 5. Finally, in Section 6, we present our conclusions and a discussion of the next steps towards an actual implementation.

2. RELATED WORK

2.1 TFRC and DCCP

The TCP-Friendly Rate Control [1] mimics the expected rate of TCP under the same network conditions. It is not a transport protocol *per se*, but can be used to implement congestion control within protocols such as DCCP.

At the beginning of a session, TFRC replicates TCP's slow-start to quickly adapt to the current network capacity. When a packet loss is first reported in the receiver's feedback messages, TFRC enters the congestion avoidance phase. In this phase, the sender adapts its rate to an estimation of that of TCP under identical network conditions using

$$X_{\mathrm{Bps}} = TFRC(p,R) = \frac{s}{R\sqrt{\frac{2p}{3}} + t_{\mathrm{RTO}}\sqrt{\frac{27p}{8}}p(1+32p^2)}, \tag{1}$$

where p is an estimate of the loss event rate, t_{RTO} is the TCP retransmission timeout (usually, 4 RTTs), and s is the packet size. The sending rate also depends on the receiver rates X_{recv}, as per (2).

$$X = \min\left(X_{\mathrm{Bps}}, 2X_{\mathrm{recv}}\right) \tag{2}$$

The Datagram Congestion Control Protocol [5] is designed to provide "congestion control without reliability" for transporting datagram-based connections. As DCCP can use TFRC for congestion control, this combination is an interesting solution to transport TCP-fair real-time traffic over the internet.

2.2 Freeze-TCP

Freeze-TCP [2] proposes to use the TCP *receiver window* header option to temporarily suspend sender activity by artificially reporting it to be null. This can be used in contexts where disconnections are predictable (*e.g.* mobile handovers or Delay Tolerant Networks) to prevent the transmission of packets bound to be lost. Disconnections can be predicted by *e.g.* monitoring the wireless signal strength or communicating with the handoff system. When reconnected, the mobile node can unfreeze the sender by re-advertizing its actual receiver window. The transmission then restarts at the previous rate instead of entering a performance-degrading slow-start phase.

3. DISCONNECTED TFRC ANALYSIS

This section introduces a model of TFRC's behavior when a disconnection occurs based on the current standard [1]. It is used to derive the number of lost packets after the disconnection as well as the delay before resuming the transmission and the resulting waste of bandwidth after a reconnection. Table 1 gives a summary of the symbols used throughout this section.

Symbol	**Meaning**
R	Sender's estimation of the RTT
X_{recv}	Sender's estimation of the receiver rate
s/t_{mbi}	Smallest allowed rate (1 packet per 64 s)
T^i_{NFI}	Start time of NFI i
X^i	Sender rate during NFI i ($X_d = X^0$)
i_x	NFI during which X^i drops below s/t_{mbi}
t^i_{RTO}	Duration of NFI i
i_t	NFI during which t^i_{RTO} starts increasing
n_{lost}	Number of packets lost during the disconnection
t_{idle}	Time before the first packet is sent
$t_{\mathrm{ss,grow}}$	Times to adapt to the new bandwidths
$n^\star_{\mathrm{wasted}}$	Number of packets that could have been sent

Table 1: List of notations used for the analysis of TFRC over a disconnection.

3.1 Evolution of Internal Parameters

Both the sender rate X and the nofeedback timer period t_{RTO} have an impact on the number of lost packets and the rate recovery after reconnecting. Just before the disconnection occurs at T_d, the sending rate is X_d. The following assumes X_d is the nominal TFRC rate the underlying link can support. Consequently, the receiver measures and reports an X_{recv} roughly equal to X_d and (2) is limited by X_{Bps} in the next computation of the sending rate.

For simplicity, the disconnected period has been segmented into *No Feedback Intervals* (NFI). An NFI is the interval between two consecutive expirations of the nofeedback timer. The first expiration marks the end of NFI 0. The rate starts decreasing at NFI 1. Similarly t_{RTO} is updated to always cover at least the emission of two packets (Figure 1(a)).

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

$$X^i = \begin{cases} \frac{X_d}{2^i} & \text{if } 0 \le i < i_x, \\ \frac{s}{t_{\mathrm{mbi}}} & \text{otherwise,} \end{cases} \tag{3}$$

$$i_x = \left\lceil \log_2 \frac{X_d \cdot t_{\mathrm{mbi}}}{s} \right\rceil. \tag{4}$$

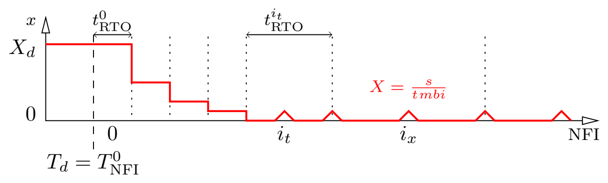
The nofeedback timer, initially set to $4R$, increases when the sending rate becomes smaller than $2s/4R$. Assuming $X_d \ge 2s/4R$ and taking i_t as the NFI during which $2s/X^{i_t}$ becomes larger than $4R$, the duration of NFI i is then

$$t^i_{\mathrm{RTO}} = \begin{cases} 4R & \text{if } i < i_t, \\ \frac{2s}{X^i} & \text{otherwise,} \end{cases} \tag{5}$$

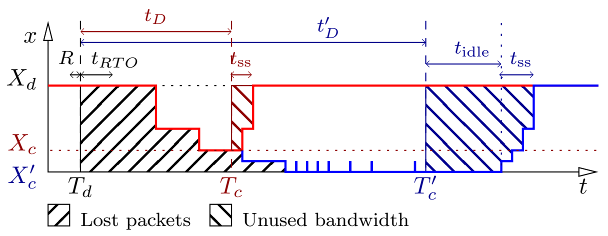
$$i_t = \left\lceil \log_2 \frac{2R \cdot X_d}{s} \right\rceil \le i_x. \tag{6}$$

3.2 Packet Losses

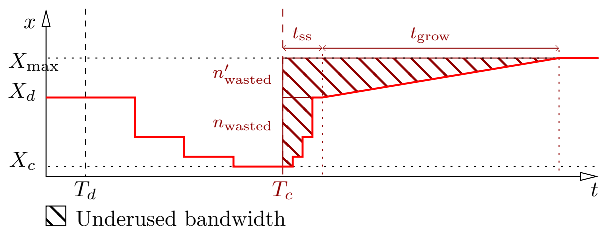
Figure 1(b) shows two cases of the evolution of the rate over a disconnection, depending the reconnection time. T_c and T'_c are respectively before and after the sender's estimation of



(a) Increase of the nofeedback timer period.



(b) Evolution of the rate over a disconnection (two cases).



(c) Slow adaptation to better network conditions.

Figure 1: Impact of a disconnection on TFRC.

X_{recv} drops to less than s/R. In the following, $t_D = T_c - T_d$ is the disconnected period, during which sent packets are lost.

3.2.1 *Estimation of the Number of Losses*

The number of lost packets after n_D NFIs (such that $\sum_{i=0}^{n_D} t^i_{\mathrm{RTO}} \geq t_D$) can be estimated using

$$n_{\mathrm{lost}} = \begin{cases} \left\lfloor \frac{7}{8}\frac{t_D X^0}{s} \right\rfloor & \text{if } t_D \leq t^0_{\mathrm{RTO}}, \\ \left\lfloor \frac{7}{8}\frac{t^0_{\mathrm{RTO}} X^0}{s} + \sum_{i=1}^{i_D-1} \frac{t^i_{\mathrm{RTO}} X^i}{s} \right. \\ \qquad \left. + \frac{t^{i_D}_{\mathrm{RTO}} X^{i_D}}{2s} \right\rfloor & \text{otherwise,} \end{cases} \tag{7}$$

where $i_D = n_D - 1$ is the index of the n_D^{th} NFI.

3.2.2 *Influence on the Loss Event Rate*

The losses will only be noticed by the receiver after reconnecting. Multiple losses during one disconnection will be considered part of a unique loss event starting in the middle of the period. The TFRC receiver computes $1/p$ as the weighted average of the last n (usually 8) loss intervals. Assuming a monotonous history of intervals of size $1/p_{\mathrm{prev}}$, the average variation can be estimated as

$$\Delta p(n_{\mathrm{pkts}}, p_{\mathrm{prev}}) = \min\left(0, \frac{\sum_{i=0}^{n-1} W_i}{W_0 n_{\mathrm{pkts}} + \frac{\sum_{i=1}^{n-1} W_i}{p_{\mathrm{prev}}}} - p_{\mathrm{prev}}\right). \tag{8}$$

3.3 Unused Bandwidth

When connectivity is re-established, two factors can cause the TFRC sender not to fully use the available bandwidth instantaneously. First the sending rate has been gradually reduced and a packet may not be sent immediately. Secondly, when feedback is received, the sending rate is not resumed directly but through a slow-start phase.

Additionally, the sender rate will be constrained by the loss event rate p which, due to its history, will reflect the old network capacity. Thus, a better network will not be used at its best until enough loss intervals have been observed.

3.3.1 *Delay Before Resuming Full Rate*

If the sending rate $X_c = X^{n_D}$ is small, the delay s/X_c between the emission of two subsequent packets becomes significant. It can take up to this delay before the first packet is sent after reconnection. An average of this idle time can be estimated as

$$t_{\mathrm{idle}} = \frac{s}{2X_c}. \tag{9}$$

After this delay, the sender starts increasing the rate, beginning at X_c. Every RTT, feedback is received with the current value of X_{recv}. The rate can then be updated to twice this value as per (2). Until X reaches the previous rate $X_d = X_{\mathrm{Bps}}$, TFRC does not perform as well as it could. The average number of packets that could additionally be sent is

$$n_{\mathrm{wasted}} = \frac{1}{s}\left(t_{\mathrm{idle}} \cdot X_d + \sum_{i=0}^{n_{\mathrm{ss}}} R_{\mathrm{new}}\left(X_d - 2^i X_c\right)\right). \tag{10}$$

Parameter n_{ss}, in (10), is such that $2^{n_{\mathrm{ss}}} X_c \geq X_d$. Consequently, time t_{ss} to re-establish the packet rate to its previous value can be expressed as

$$t_{\mathrm{ss}} = R_{\mathrm{new}} \cdot n_{\mathrm{ss}} = R_{\mathrm{new}} \left\lceil \log_2 \frac{X_d}{X_c} \right\rceil. \tag{11}$$

3.3.2 *Network with a Larger Bandwidth*

The estimate of the loss event rate p is designed to evolve smoothly. When the new network conditions are better, it may take an unacceptably long time for the sender to use the full available capacity (Figure 1(c)), resulting in more wasted bandwidth, as per

$$\begin{aligned} n'_{\mathrm{wasted}} = {} & \frac{1}{s}(X_{\mathrm{max}} - X_d)(t_{\mathrm{idle}} + t_{\mathrm{ss}}) \\ & + \frac{R_{\mathrm{new}}}{s} \sum_{i=0}^{n_{\mathrm{grow}}} \left(X_{\mathrm{max}} - X^i\right) \end{aligned} \tag{12}$$

with

$$X^i = \begin{cases} X_d & \text{if } i = 0, \\ \min\left(X_{\mathrm{Bps}}\left(p_{\mathrm{r}} + \Delta p(n^{i-1}_{\mathrm{pkts}}, p_{\mathrm{r}}), R_{\mathrm{new}}\right), \right. \\ \qquad \left. 2X^{i-1}\right) & \text{otherwise,} \end{cases}$$

$$n^i_{\mathrm{pkts}} = \frac{1}{s}\left(R_{\mathrm{new}} \sum_{j=0}^{i} X^j\right).$$

Similarly to (10), n_{grow} is the number of RTTs needed to have $X^{n_{\mathrm{grow}}} \geq X_{\mathrm{max}}$. The equivalent loss event rate p_r is such that $TFRC(p_r, R_{\mathrm{new}}) = X_d$.

Table 2: Packet losses and unused bandwidth 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
Unused bandwidth [500 B packets]				
UMTS	0	82938	263	109541
802.16	0	471	155	1029
802.11b	0	0	1085	54674
802.11g	0	0	0	4699

3.4 Validity of the Model

The model has been validated against ns-2 simulations. Due to space constraints, more specific details had to be omitted.[1] The reader can however verify that predicted performances in Table 2 adequately match that of DCCP/TFRC simulations in Table 3, Section 5.1. Comparison of additional analytical and simulation results indicates a sufficient prediction accuracy for use in estimating potential performance gains.

4. PROPOSED SOLUTION: FREEZE THE DCCP/TFRC CONNECTION

Performance improvements that could be expected from a better handling of handovers have been derived for Mobile IPv6 in realistic roaming scenarios. The results presented in Table 2 clearly show that the behavior of DCCP/TFRC can be improved. In this section, we present an enhancement to mitigate the observed sub-optimal performance.

To avoid sending packets bound to be lost, the sender blocks the evolution of specific parameters and suspends its transmission. As DCCP does not provide reliability, new packets are simply discarded. When connectivity is available anew, the rate is restored and the congestion control algorithm allows packets to be sent at the same rate as before. If no error is reported, the sender then tries to probe the network to adapt faster to higher capacities.

Overall, the Freeze-DCCP/TFRC operation is separated into three phases: *Frozen*, *Restoring* and *Probing*. New states are implemented into the sender and receiver to support these. Additionally, new DCCP options are introduced to enable the required signaling for state transition and synchronization.

Figure 2 shows the proposed Freeze-DCCP state diagram. The sender has three new states, shown in Figure 2(a). As most of Freeze-DCCP's operation is driven by the sender, they 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 synchronization with the sender. These are shown in Figure 2(b).

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

4.1 Additional Signaling

TFRC not having the concept of a receiver window, it is not possible to freeze and unfreeze a DCCP/TFRC sender as conveniently as it was in Freeze-TCP. Also, it is desirable to be able to locally suspend the sender. To fully support freezing on both sides, several new signaling options have to be introduced, to be carried in the DCCP packet header. As DCCP ignores unknown options, compatibility with standard implementations is retained.

When an upcoming disconnection is predicted, a mobile node will send packets with an `OPT_FREEZE` option to inform the remote peer's sender. It will then freeze locally. When connectivity becomes available again, the node can restart its traffic, and inform the other end using the `OPT_UNFREEZE` option.

Additional options are used to support further signaling during the unfreezing phases. The sender uses `OPT_PROBING` and `OPT_RESTORING` to indicate the state that it is currently in, 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 synchronized. 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 risks consuming too much bandwidth and reduction of the frequency could be considered.

4.2 Frozen Phase

When instructed to freeze the sender enters the Frozen state. In this state, all data emission ceases, which ensures that no packet will be lost. The loss event rate calculated by the receiver will thus be kept unmodified. The receiver doesn't need any specific state to support this phase.

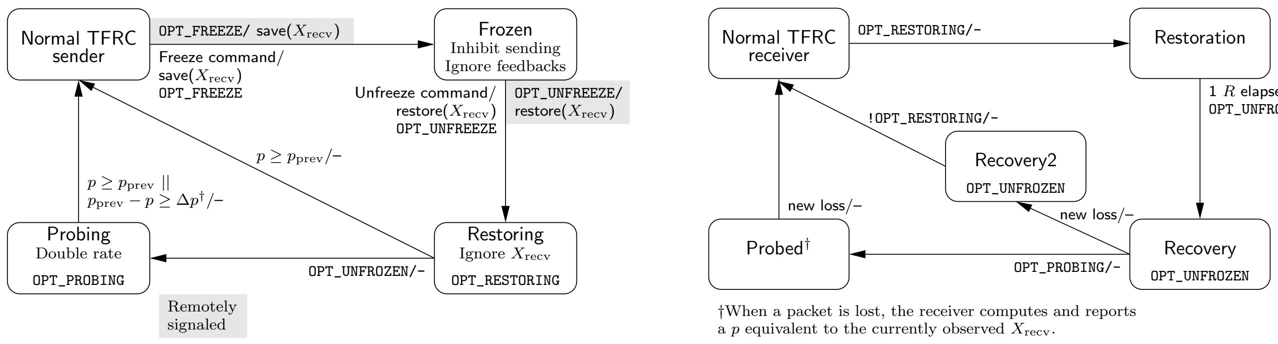
The disconnection may, however, not happen right after freezing and additional feedback from the receiver may arrive at the sender. Parameters such as the RTT R, or the receiver rate X_{recv} risk being updated. Thus the sender has to ignore all feedback. When entering the frozen state, it also saves the value of X_{recv} as it will be locally modified with expirations of the nofeedback timer.

To efficiently address longer disconnection periods which may occur (*e.g.* in DTNs), it is advisable to additionally increase the connection timeout.

4.3 Restoring Phase

After receiving an unfreeze instruction (or an `OPT_UNFREEZE` option), the sender will enter the Restoring state. It first restores X_{recv}. The send timer is then reset to resume the

[1]More details about the simulations parameters and scenarios mentioned throughout this paper can be found at http://www.nicta.com.au/people/mehanio/freezedccp#simulations.


Normal TFRC sender
OPT_FREEZE/ save(X_{recv})
Freeze command/ save(X_{recv}) OPT_FREEZE
Frozen
Inhibit sending
Ignore feedbacks
Unfreeze command/ restore(X_{recv}) OPT_UNFREEZE
OPT_UNFREEZE/ restore(X_{recv})
$p \geq p_{\text{prev}}$/–
$p \geq p_{\text{prev}}$ || $p_{\text{prev}} - p \geq \Delta p^\dagger$/–
Probing
Double rate
OPT_PROBING
OPT_UNFROZEN/–
Restoring
Ignore X_{recv}
OPT_RESTORING
Remotely signaled
Normal TFRC receiver
OPT_RESTORING/–
Restoration
1 R elapsed/ OPT_UNFROZEN
!OPT_RESTORING/–
Recovery2
OPT_UNFROZEN
new loss/–
new loss/–
Probed†
OPT_PROBING/–
Recovery
OPT_UNFROZEN
†When a packet is lost, the receiver computes and reports a p equivalent to the currently observed X_{recv}.

(a) Freeze-DCCP/TFRC sender (b) Freeze-DCCP/TFRC receiver

Figure 2: Additional states and options exchanges to support Freeze-DCCP/TFRC (transitions are labeled 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.

packets transmission. As the parameters are the same as before the disconnection, the previous sending rate will be restored.

At the same time, feedback from the receiver is no longer ignored, except for X_{recv} reports. Indeed, this value has to be measured over at least one RTT. The first feedbacks 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, bound by X_{recv} as per equation (1).

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

4.4 Probing Phase

Standard TFRC already provides for a reduction in the available bandwidth by responding quickly to an increase in the loss event rate but has no mechanism to quickly adapt to better network conditions. In the Probing state, entered if the loss event rate hasn't increased during the Restoring phase, the Freeze-DCCP sender checks for such improvement. It uses the `OPT_PROBING` option to inform its peer. Upon reception of this option, the receiver enters the Probed state.

This phase is similar to a slow-start. Every RTT, the sending rate is doubled. When it detects a loss, the receiver first computes a p equivalent to the last observed received rate. It then reinitializes its loss history to match calculated size and reports this value to the sender.

The criteria for the sender to exit the Probing state are based on this reported loss event rate. In a loss-less period, p will never increase and keep decreasing slightly. The sender should thus exit the Probing state if $p > p_{\text{prev}}$ or $p_{\text{prev}} - p > \Delta p$ (Δp is arbitrarily chosen as $0.01 p_{\text{prev}}$). The absence of

Table 3: Simulated MIPv6 handovers performance impact for DCCP/TFRC (top cell) and Freeze-DCCP/TFRC (bottom cell).

to / from	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
Unused bandwidth [500 B packets]				
UMTS	50.5	54018.05	2209.5	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

the `OPT_PROBING` option on new packets will in turn take the receiver out of the Probed state.

5. PERFORMANCE EVALUATION

The enhancements proposed in section 4 have been implemented within ns-2. Comparison of Freeze-DCCP/TFRC performance with the regular version, as well as preliminary fairness assessment, are shown here.

5.1 Realistic Handovers Scenarios

Figure 3 shows a comparison of how both regular DCCP and the Freeze-enabled version perform in key example scenarios.

The number of packet losses and unused bandwidth upon reconnection is shown on Table 3. The unused bandwidth has been estimated by comparing TFRC's actual X to what is achievable in the steady state then converted in number of packets. As Freeze-DCCP/TFRC did not lose any packet,

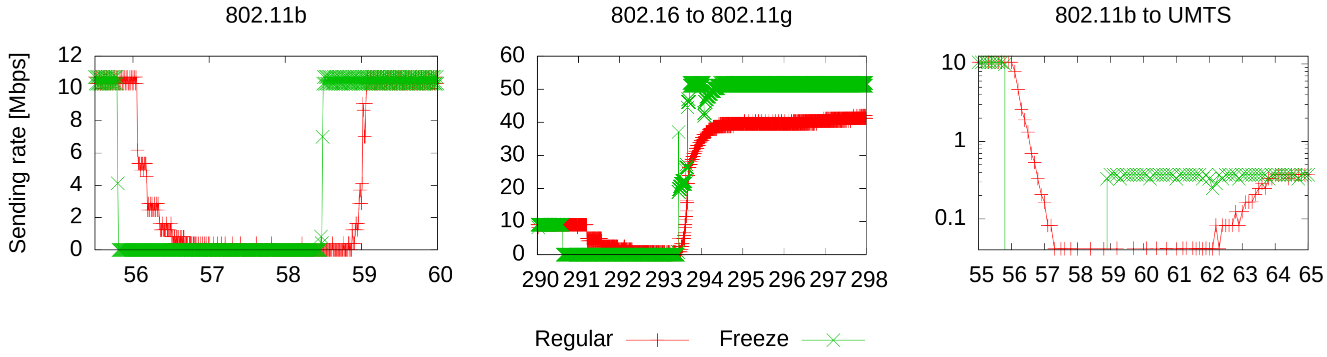


Figure 3: Comparison of the rate of DCCP/TFRC and the Freeze-enabled version in example MIPv6 horizontal or vertical handovers.

Table 4: Fairness of Freeze-DCCP to TCP after a handover. Values in the range $[0.5, 2]$ are considered "reasonably fair" [1].

from \ to	UMTS	802.16	802.11 b	802.11 g
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

this information has been omitted.

Freeze-DCCP successfully avoids losses during the disconnections. Additionally, the under-usage of the bandwidth is greatly reduced.

5.2 Fairness to TCP Flows

TFRC was designed to quickly respond to reductions of the bandwidth. The restoring and probing features of Freeze-DCCP/TFRC, however, aggressively use and test the network. It is important to check that these additions do not make the protocol too greedy, breaking TFRC's important property of being fair to TCP. The criterion to evaluate TCP-fairness is the bandwidth occupation ratio of Freeze-DCCP to concurrent TCP flows.

Table 4 shows the average fairness of a Freeze-DCCP flow to a concurrent TCP stream, as observed after the reconnection for the studied handover scenarios. The proposed improvement has been found to retain the original TFRC's TCP-fairness property in various simulated scenarios.

6. CONCLUSIONS AND FUTURE WORK

We have analytically derived the losses and subsequent under-usage of the available bandwidth that TFRC experiences in mobility scenarios. This showed what improvements could be expected from a system with a better awareness and handling of disconnections. We thus proposed Freeze-DCCP, an extension of the TFRC congestion control mechanism for DCCP. Our proposal is aimed at uses of DCCP in situations where network connectivity may periodically not be available for varying periods of time.

Simulation results have shown that it was possible to prevent disconnection-induced losses, to restore the rate and adapt faster to higher capacity networks upon reconnection, without losing TCP-fairness. We argue that the proposed extension to DCCP can significantly improve real-time performance when disconnections are predictable, particularly for IP mobility or, more generally, Delay Tolerant Networks.

Next steps include implementing the proposal in a real system and confirming the encouraging simulation results in real experiments. Future work will also cover mechanisms to enable abstract information exchange between layers, as was assumed to be available in this paper.

7. ACKNOWLEDGMENTS

This research work has been supported by funding from National ICT Australia Ltd. The authors would like to thank Guillaume Jourjon for his most valuable advice and proofreading help, as well as the anonymous reviewers for their insightful comments.

8. REFERENCES

[1] S. Floyd, M. Handley, J. Padhye, and J. Widmer. TCP friendly rate control (TFRC): Protocol specification. RFC 5348 (Proposed Standard), Sept. 2008.

[2] T. Goff, J. Moronski, D. S. Phatak, and V. Gupta. Freeze-TCP: A true end-to-end TCP enhancement mechanism for mobile environments. In *IEEE INFOCOM '00*, 2000.

[3] A. Gurtov and J. Korhonen. Effect of vertical handovers on performance of TCP-friendly rate. *ACM SIGMOBILE: Mobile Computing Communications Review*, 8, 2004.

[4] D. Johnson, C. Perkins, and J. Arkko. Mobility support in IPv6. RFC 3775 (Proposed Standard), June 2004.

[5] E. Kohler, M. Handley, and S. Floyd. Designing DCCP: Congestion control without reliability. In *ACM SIGCOMM '06*, Sept. 2006.

[6] V. Tsaoussidis and I. Matta. Open issues on TCP for mobile computing. *Wireless Communications & Mobile Computing*, 2(1), Feb. 2002.