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 reconections 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 ana-
lytical 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 the actual receiver window. The transmission then restarts at the sender rate during NFI i (X_i = X_{recv}^i). The nofeedback timer, initially set to 4s, increases when the TCP retransmission timeout (usually, 4 RTOs), and s is the sending rate also depends on the receiver rates X_{recv}, as per (2).

\[ X = \min(X_{bps}, 2X_{recv}) \]  
(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 handheld system. When reconnected, the mobile node can unfreeze the sender by re-advertising 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.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X )</td>
<td>Sender’s estimation of the RTT</td>
</tr>
<tr>
<td>( X_{recv} )</td>
<td>Sender’s estimation of the receiver rate</td>
</tr>
<tr>
<td>( s/t_{mbi} )</td>
<td>Smallest allowed rate (1 packet per 64s)</td>
</tr>
<tr>
<td>( t_{NFI} )</td>
<td>Start time of NFI i</td>
</tr>
<tr>
<td>( X^i )</td>
<td>Sender rate during NFI i (( X_d = X^0 ))</td>
</tr>
<tr>
<td>( i_x )</td>
<td>NFI during which ( X^i ) drops below ( s/t_{mbi} )</td>
</tr>
<tr>
<td>( t_{NFI} )</td>
<td>Duration of NFI i</td>
</tr>
<tr>
<td>( i_t )</td>
<td>NFI during which ( t_{NFI} ) starts increasing</td>
</tr>
<tr>
<td>( n_{lost} )</td>
<td>Number of packets lost during the disconnection</td>
</tr>
<tr>
<td>( t_{idle} )</td>
<td>Time before the first packet is sent</td>
</tr>
<tr>
<td>( t_{supported} )</td>
<td>Times to adapt to the new bandwidths</td>
</tr>
<tr>
<td>( n_{wasted} )</td>
<td>Number of packets that could have been sent</td>
</tr>
</tbody>
</table>

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 \( t_{NFI} \) 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 \( t_{NFI} \) 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 \( 2X_{recv}^i \) drops below \( s/t_{mbi} \), the sender rate can be expressed as

\[ X^i = \begin{cases} \frac{X_d}{2} & \text{if } 0 \leq i < i_x, \\ \frac{X_d}{2} & \text{otherwise}, \end{cases} \]  
(3)

\[ i_x = \left\lfloor \log_2 \left( \frac{X_d \cdot t_{mbi}}{s} \right) \right\rfloor. \]  
(4)

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

\[ t_{NFI} = \begin{cases} \frac{4R}{X_d} & \text{if } i < i_t, \\ \frac{X_d}{s} & \text{otherwise}, \end{cases} \]  
(5)

\[ i_t = \left\lfloor \log_2 \left( \frac{2R \cdot X_d}{s} \right) \right\rfloor \leq i_x. \]  
(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
average variation can be estimated as a weighted average of the last p packets of the period. The TFRC receiver computes 1/p under the assumption that each lost packet is considered part of a unique loss event starting in the middle of the period. Let \( NFI \) be the number of lost packets after \( n \) RTTs.

\[ n_{\text{lost}} = \begin{cases} \left\lfloor \frac{7}{8} \frac{X^0}{s} \right\rfloor & \text{if } t_D \leq t_{\text{RTT}}^0, \\ \frac{7}{8} \frac{X^0}{s} + \frac{t_{\text{RTT}}^0 X^i - 1}{8} & \text{if } t_D \leq t_{\text{RTT}}^0, \\ \frac{t_{\text{RTT}}^0 X^i + X'^i}{2s} & \text{otherwise,} \end{cases} \]

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_{prev}, the average variation can be estimated as

\[
\Delta p(n_{\text{pktls}}; p_{\text{prev}}) = \min \left(0, \frac{\sum_{i=1}^{n-1} W_i}{W_0 n_{\text{pktls}} + \sum_{i=1}^{n-1} W_i \frac{p_{\text{prev}}}{}^\text{prev}} - p_{\text{prev}} \right).
\]

### 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 \( X_c = X^{n_{\text{RTT}}} \) 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

\[ t_{\text{idle}} = \frac{s}{2X_c}. \]

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

\[ n_{\text{wanted}} = \frac{1}{s} \left( t_{\text{idle}} \cdot X_d + \sum_{i=0}^{n_{\text{grow}}} R_{\text{new}} \left( X_d - \frac{2^i X_c}{} \right) \right). \]

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

\[ t_{\text{srt}} = R_{\text{new}} \cdot n_{\text{srt}} = R_{\text{new}} \left( \log_2 \frac{X_d}{X_c} \right). \]

#### 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 unacceptable long time for the sender to use the full available capacity (Figure 1(c)), resulting in more wasted bandwidth, as per

\[ n_{\text{wanted}} = \frac{1}{s} \left( X_{\text{max}} - X_c \right) \left( t_{\text{idle}} + t_{\text{srt}} \right) + R_{\text{new}} \sum_{i=0}^{n_{\text{grow}}} \left( X_{\text{max}} - X^i \right) \]

with

\[ X^i = \begin{cases} X_d & \text{if } i = 0, \\ \min \left( X_{\text{Bstmt}}, p_t + \Delta p \left(n_{\text{pktls}}(p_t), R_{\text{new}} \right), R_{\text{new}} \right) & \text{otherwise}, \end{cases} \]

\[ n_{\text{pktls}} = \frac{1}{s} \left( R_{\text{new}} \sum_{i=0}^{n_{\text{grow}}} X^i \right). \]

Similarly to (10), \( n_{\text{grow}} \) is the number of RTTs needed to have \( X^{n_{\text{grow}}} \geq X_{\text{max}} \). The equivalent loss event rate \( p_t \) is such that \( \text{TFRC}(p_t, R_{\text{new}}) = X_c \).
Table 2: Packet losses and unused bandwidth expected during a MIPv6 handover.

<table>
<thead>
<tr>
<th></th>
<th>from UMTS</th>
<th>802.16 b</th>
<th>802.11 g</th>
</tr>
</thead>
<tbody>
<tr>
<td>Packet losses</td>
<td>306</td>
<td>236</td>
<td>226</td>
</tr>
<tr>
<td></td>
<td>2760</td>
<td>2614</td>
<td>2614</td>
</tr>
<tr>
<td></td>
<td>1080</td>
<td>1078</td>
<td>1078</td>
</tr>
<tr>
<td></td>
<td>2909</td>
<td>2907</td>
<td>2907</td>
</tr>
<tr>
<td>Unused bandwidth [500 B packets]</td>
<td>0</td>
<td>82938</td>
<td>263</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>471</td>
<td>155</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>1085</td>
</tr>
<tr>
<td></td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

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_RESTORED 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 transmission 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 \( X_{rec} \) risk being updated. Thus the sender has to ignore all feedback. When entering the frozen state, it also saves the value of \( X_{rec} \) 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 \( X_{rec} \). The send timer is then reset to resume the
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 $X_{\text{recv}}$ 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 $X_{\text{recv}}$. Using such value may create instabilities in the sending rate, bound by $X_{\text{recv}}$ 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 $X_{\text{recv}}$.

### 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 $p > p_{\text{recv}}$ or $p_{\text{recv}} - p > \Delta p$ $(\Delta p$ is arbitrarily chosen as $0.01p_{\text{recv}})$. The absence of 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,
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 proof-reading help, as well as the anonymous reviewers for their insightful comments.

### 8. REFERENCES


