Mobile Multimedia Streaming Improvements with Freeze-DCCP*

Olivier Mehani[†,‡,§,¶] Roksana Boreli[†,‡] Guillaume Jourjon[†] Thierry Ernst[¶]

[†]National ICT Australia (Nicta)
Locked Bag 9013
Alexandria, NSW 1435,
Australia
firstname.lastname@nicta.com.au

[¶]Inria Rocquencourt
Domaine de Voluceau, BP 105
78153 Le Chesnay, France
firstname.lastname@inria.fr

[‡]University of New South Wales
Sydney, Australia

[§]Mines ParisTech
Paris, France

ABSTRACT

We propose to demonstrate a cross-layer enhancement of DCCP/TFRC which allows the transport to better handle network mobility handovers and adapt faster to the capacity available in the new network. We argue Freeze-DCCP is well suited for real-time traffic such as multimedia streaming. The mechanism has been implemented in the Linux kernel and the demonstration is run in a mobility testbed.

Categories and Subject Descriptors

C.2.5 [**Computer Systems Organization**]: Computer-Communication Networks—*Local and Wide-Area Networks*; C.2.6 [**Computer Systems Organization**]: Computer-Communication Networks—*Internetworking*

General Terms

Algorithms, Experimentation, Performance

Keywords

TFRC, DCCP, mobile streaming, congestion control

1. INTRODUCTION

Recent years have seen the increase of lightweight mobile devices with multiple wireless connectivity options. With the ever increasing computation power of these devices, more and more on-line services are now easily accessible on the go from these mobile terminals. Multimedia streaming, either unidirectional in the case of program broadcasts or bidirectional in the case of audio- and video-conferencing, is becoming a prevalent area of interest for mobile applications.

To support such data streaming on a large scale without risking a global congestion collapse, it is important that the protocols used to transport these real-time streams are aware of the congestion levels and able to adapt their rate to share the network fairly. The two classical transport protocols are currently used for real-time data streaming on shared networks. However, they are not very well suited for such a purpose. UDP simply doesn't assess, let alone adapt to, the congestion levels while TCP's retransmissions used to ensure reliability works against the timing requirements of multimedia traffic.

Several transport protocols have thus been proposed to provide congestion-controlled datagrams, trading off reliability for timeliness. In 2006, Kohler *et al.* introduced the Datagram Congestion Control Protocol [6]. One of the advantages of DCCP is its support for "pluggable congestion control," letting the user choose the most appropriate algorithm for their applications. The most commonly used for multimedia traffic is the TCP-Friendly Rate Control [1], an equation-based congestion control protocol which mimics TCP's rate behaviour in the same network conditions. It is available to DCCP as *Congestion Control ID 3* [2]. Using DCCP/TFRC is thus a very promising option for real-time multimedia streaming.

However, when a mobile device switches from one network to another, *e.g.* in Mobile IPv6 [5] handovers, it usually experiences short periods of disconnection [7] during which datagrams are lost. These losses are erroneously considered an indication of congestion by the transport, and the data rate is unnecessarily reduced [3]. In the worst case scenario, a TFRC sender may even reduce its rate whereas the new network could actually support a higher data rate, than the previous one. This rate reduction, and the subsequent time to increase it to the network's capacity, is detrimental to the multimedia streaming quality as it is not possible for the application to send datagrams at the rate required by the codec, resulting in packet or frame losses. Some solutions exist to avoid losing packets during a hand-off but, as they rely on packet buffering, they are not well suited for real-time traffic.

*Nicta is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

Demonstration presented at *MobiCom'10* and the co-located *WiNTECH'10* workshop Chicago, IL, USA.

In [8], we proposed cross-layer additions to DCCP's implementation of TFRC to mitigate the impact of mobile handovers. Based on notifications from the mobility system, a sender or a receiver can be instructed to *freeze* the data stream temporarily, then reestablish it at the previous rate and probe for higher bandwidths. We showed in simulation that these enhancements effectively limit the impact on the overall quality of the multimedia stream by letting the sender discover and use the network capacity available after a handover much faster than the standard TFRC.

We propose to demonstrate these enhancements, as implemented within the Linux kernel, in a mobility testbed. The rest of this text outlines Freeze-DCCP/TFRC in Section 2, presents the demonstration scenario and details some technical aspects in Section 3.

2. OVERVIEW OF FREEZE-DCCP/TFRC

Freeze-DCCP's enhancements are primarily additions to the TFRC algorithm. New states have been added to both the TFRC sender and receiver (Fig. 1) so they can synchronise with each other. Through the use of additional DCCP options, either end can suspend the entire connection. This way, the stream can be suspended even if the sender is not the mobile endpoint. In the following, we outline the behaviour of a single *half-connection*, but this can be generalised for its symmetrical counterpart.

Upon reception of either a trigger from the mobility system or a DCCP packet with the `OPT_FREEZE` option from the other end, the TFRC sender saves the current value of two parameters used in the rate calculation: the loss event rate p and the receiver rate X_{recv}. It then suspends its transmission as long as it is in the *Frozen* state (Fig. 1(a)). The receiver doesn't need to take any special action at this stage.

After receiving an indication that the handover is finished, the sender restores the saved parameters and enters the *Restoring* phase during which it signals the receiver using the `OPT_RESTORING`. From then on, the rate restoration is driven by the receiver (Fig. 1(b)). The sender transmits datagrams at the same rate as before, while the receiver, in the *Restoration* phase, measures the received bandwidth. If any loss is experienced, normal TFRC operation is resumed.

If no loss has occurred during one RTT of the Restoring/Restoration phase, the receiver signals the sender using the `OPT_UNFROZEN` option. The sender then enters the *Probing* phase, which is similar to a slow-start. The probing phase is terminated after the first loss is detected. The receiver recomputes a loss event rate reflecting the available bandwidth and transmits it to the sender. As p has increased or varied by more than Δp (the maximal "normal" variation of p), the sender then reverts back to standard TFRC operation.

This new mechanism thus allows the transport to benefit from mobility information to 1) not disrupt its rate because of non-congestive losses during a handover 2) recover immediately and 3) probe for more bandwidth if it can benefit the application quality.

3. DEMONSTRATION

3.1 Scenario

To demonstrate the performance improvements of our proposal, we use the Nicta's Orbit testbed for network experiments to emulate vertical handovers and observe the impact on the quality of a video stream.

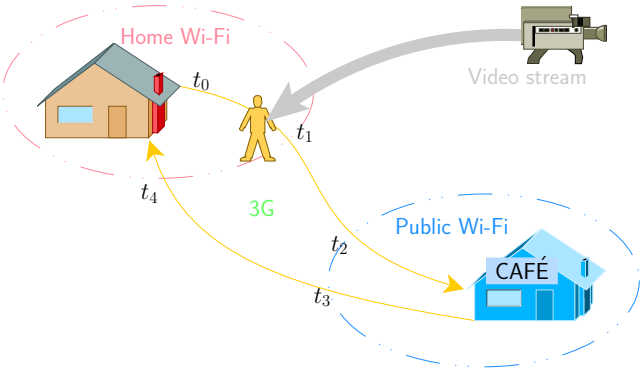


Figure 2: Demonstration scenario: a user viewing a video stream on their mobile device goes out for a coffee then comes back home. In the process, different networks with various capacities are visited.

We consider a common scenario when a user, initially at home (t_0), receives a video stream on their mobile terminal connected to their home Wi-Fi network, as shown in Fig. 2. 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 the 3G network (t_1). The coffee shop has a public wireless network, to which the device connects when it gets 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 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 to various wireless networks.

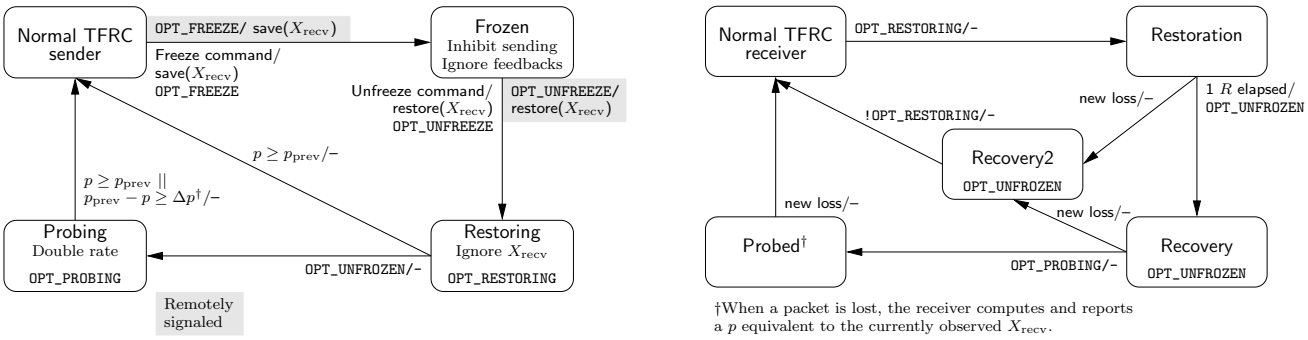
Our enhancement is compared to standard DCCP/TFRC, with the mobile receiver controlling when to freeze and unfreeze (using the newly introduced DCCP options) in the first case, to show the performance improvement. To this end, metrics such as the sending and receiving rates at the transport level and video quality (as quantified by its peak signal-to-noise ratio) at the application level are monitored and displayed in real time.

3.2 What's under the hood

Freeze-DCCP has been implemented in the Linux kernel on top of the DCCP development code base.[1] The patched kernel has been installed on the NOrbit nodes used as sender and receiver.

A video file, encoded and packetised using the H.264 codec with a 1Mbps bit rate, is sent to a custom receiver application using a specially patched version of Iperf.[2] 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). Both sender and receiver have been instrumented with the OML library [10]. This allows them to report readings of these metrics in real time for analysis or display.

[1]Linux 2.6.34-r5, see http://www.nicta.com.au/people/mehanio/freezedccp#linux26.
[2]http://www.nicta.com.au/people/mehanio/freezedccp#iperf-dccp-oml



(a) Freeze-DCCP/TFRC sender

(b) Freeze-DCCP/TFRC receiver

Figure 1: Additional states and options 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.

In order to emulate handovers in the presented scenario, we are using the cOntrol and Management Framework for testbeds (OMF) [9]. It allows an experimenter to describe the entire experiment with an OEDL script.[3] Furthermore, we have extended the OEDL library to allow the dynamic reconfiguration of the topology. The handovers are emulated thanks to the instrumentation NetEm [4] and `iptables`.

More details on this demo and OMF can be found at http://omf.mytestbed.net/projects/omf-case-studies/wiki/FreezeDccpQoE.

4. REQUIREMENTS

4.1 Equipment

As the actual demo runs on resources in our Australian-based testbed we only need an Internet connection to the OMF portal and space for a monitor to display the experiment control interface. In addition, it will be beneficial to the understanding of the demo to have a stand for a summary poster.

4.2 Eligibility for the competition

This demo is eligible for the student demo competition. Olivier Mehani from Nicta is the leading student.

5. REFERENCES

[1] S. Floyd, M. Handley, J. Padhye, and J. Widmer. Equation-based congestion control for unicast applications. *SIGCOMM Computer Communication Review*, 30(4):43–56, October 2000.

[2] 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 (Proposed Standard), March 2006.

[3] A. Gurtov and J. Korhonen. Effect of vertical handovers on performance of TCP-friendly rate control. *SIGMOBILE Mobile Computing and Communications Review*, 8(3):73+, July 2004.

[4] S. Hemminger. Network emulation with NetEm. In M. Pool, editor, *LCA 2005, Australia's 6th national Linux conference (linux.conf.au)*, Sydney NSW, Australia, April 2005. Linux Australia, Linux Australia.

[5] D. Johnson, C. Perkins, and J. Arkko. Mobility support in IPv6. RFC 3775 (Standards Track), June 2004.

[6] E. Kohler, M. Handley, and S. Floyd. Designing DCCP: Congestion control without reliability. *SIGCOMM Computer Communication Review*, 36(4):27–38, October 2006.

[7] J. S. Lee, S. J. Koh, and S. H. Kim. Analysis of handoff delay for mobile IPv6. In T. M. Nguyen, editor, *VTC2004-Fall, 60th IEEE Vehicular Technology Conference*, volume 4, pages 2967–2969 Vol. 4, Los Alamitos, CA, USA, September 2004. IEEE Computer Society.

[8] O. Mehani, R. Boreli, and T. Ernst. Analysis of TFRC in disconnected scenarios and performance improvements with Freeze-DCCP. In J. Ott and K. Tan, editors, *MobiArch 2009, 4th International Workshop on Mobility in the Evolving Internet Architecture*, New York, NY, USA, June 2009. ACM SIGMOBILE, ACM.

[9] T. Rakotoarivelo, M. Ott, G. Jourjon, and I. Seskar. OMF: A control and management framework for networking nestbeds. *SIGOPS Operating Systems Review*, 43(4):54–59, January 2010.

[10] J. White, G. Jourjon, T. Rakotoarivelo, and M. Ott. Measurement architectures for network experiments with disconnected mobile nodes. In A. Gavras, N. Huu Thanh, and J. Chase, editors, *TridentCom 2010, 6th International ICST Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities*, Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, Heidelberg, Germany, May 2010. ICST, Springer-Verlag Berlin.

[3] More details about the description language can be found at http://omf.mytestbed.net.