Mitigating the Impact of Packet Reordering to Maximize Performance of Multimedia Applications

Golam Sarwar[*,†]
[*] National ICT Australia Ltd, Australia,
[†] University of NSW, Kensington, Australia,
golam.sarwar.@nicta.com.au

Emmanuel Lochin
Université de Toulouse,
ISAE, LAAS-CNRS, France,
emmanuel.lochin@isae.fr

Roksana Boreli[*,†]
roksana.boreli.@nicta.com.au

***Abstract*—We propose a solution to mitigate the performance degradation and corresponding Quality of Experience (QoE) reduction caused by packet reordering for multimedia applications which utilise unreliable transport protocols like the Datagram Congestion Control Protocol (DCCP). We analytically derive the optimum buffer size based on the applications data rate and the maximum delay tolerated by the multimedia application. We propose a dynamically adjustable buffer in the transport protocol receiver which uses this optimum buffer size. We demonstrate, via simulation results, that our solution reduces the packet loss rate, increases the perceived bandwidth and does not increase jitter in the received applications packets while still being within the application's delay limits, therefore resulting in an increased QoE of multimedia applications.**

***Index Terms*—transport protocol; reordering; QoE; VoIP; video;**

I. Introduction and Motivation

Multimedia applications have an increasingly large share of the overall Internet traffic [1]. To combat unfair advantage and increasing Internet congestion introduced when such applications are transmitted using the UDP protocol, transport protocols which provide congestion control but no unnecessary reliability have been proposed such as DCCP [2] and SCTP [3]. DCCP regulates congestion and is well suited to VoIP and video applications which require timely delivery of application packets rather than reliability unbounded by any time limit.

Packet reordering has long been considered as a non pathological network behaviour [4]. More recent studies have supported this premise, for instance, the study in [6] finds that 70% of packets are correctly ordered taking into consideration one specific GEANT backbone and in [7], the authors show that 40% of the links present in their dataset effectively reorder packets. The reasons for having out of sequence packets at the receiver include load balancing, multiple network paths, dynamic route generation and link bonding [8]. Additionally, in mobile devices with multiple wireless network interfaces, handover between different wireless technologies may also result in out of order packets. In today's Internet the amount of out of sequence packets is about $3\% - 5\%$ [7]. It has also been shown by previous studies that in a single stream transmission, the probability of packet reordering is increased as the transmission rate increases [8].

We note that congestion control, reliability and in-order packet delivery are separate issues which can be handled by transport protocols. Reliable transport protocols such as TCP inherently include a mechanism to handle out of order packet delivery. This is done by the in built acknowledgment and retransmission mechanism and required by the applications using TCP. However, due to the lack of reliability (which is not needed), unreliable transport protocols like DCCP miss the important capability for in-order packet delivery, which has to be provided by different means. In this paper, we propose a buffer based mechanism which ensures ordered packet delivery to the multimedia applications following a given time threshold. Indeed, although multimedia applications do not need reliability, their performance could significantly decrease when they do not have an in-order delivery service. Furthermore, we show that the use of a reordering buffer also improves the performance of unreliable congestion-controlled protocol such as DCCP, which interpret out of order packets as losses, indicating congestion.

The paper is organised as follows: Section II outlines our proposal, Section III presents the details of the simulation setup, parameters and results, Section IV shows the improvements achieved in the QoE for example VoIP and video applications and we conclude in Section V.

II. Dynamic Receiver Buffer

We propose to use a receiver buffer with a dynamically adjustable size to handle the out of order received packets. We first present a rationale for our proposal, followed by the details of buffer data management. Our aim is to minimise the residual negative effects which such a buffer could have on the multimedia applications quality, i.e. the increased delay and jitter.

The proposal focuses on DCCP [2] protocol, which includes a number of options to enable the type of congestion control suited to applications requirements. As our interest is in VoIP and multimedia applications, we choose TFRC based DCCP Congestion Control IDs CCID3 [9] and CCID4 [10] to evaluate the effects of reordering. TFRC defines a rate based congestion control mechanism. After the initial slow-start like period, the sender will regulate the transmitted rate based on the receiver feedback. The receiver considers the received packet sequence numbers to detect losses and estimate the packet loss rate, indicated by the loss event rate p. This, together with the estimate of delay and the received data rate, is included in feedback packets to the DCCP sender. The sender adds the estimated receiver to sender delay to derive the return trip time and calculates the data rate for the latest reporting period by using a combination of the receiver rate and a rate calculated by the equation (1). This approach is used in order to provide fairness to TCP flows on the same link.

$$X = \frac{s}{RTT \cdot \sqrt{\frac{p \cdot 2}{3}} + RTO \cdot \sqrt{\frac{p \cdot 27}{8}} \cdot p \cdot (1 + 32 \cdot p^2)} \tag{1}$$

where: s is the packet size in bytes; p is the loss event rate; RTO is the TCP retransmission timeout value in seconds.

The generic TFRC mechanism described above is used in the CCID3 type of congestion control. CCID4 [10] differs from CCID3 in that it is adjusted to small packets sizes appropriate to VoIP. In place of the actual packet size, CCID4 uses a fixed packet size of 1460 bytes modified by a header correction factor. This ensures that the formula based rate from equation (1) which is directly proportional to the applications packet size, does not unfairly disadvantage DCCP, by using a common TCP packet size in place of the actual size of smaller VoIP packets.

It is worthwhile to highlight the effect of out or order packets on both the transport protocol mechanisms and to the applications and some differences between how TCP, UDP and DCCP handle these packets:

1) When the TCP retransmission timer (RTO) expires, TCP triggers a Go Back N recovery procedure which could lead to retransmissions of packets effectively received (i.e. spurious retransmissions). This case can occur when TCP considers losses following the reordering of network packets or a significant increase of the RTT (e.g. in case of a vertical handoff, load balancing, or networks misconfiguration). In this particular case, known as spurious timeout, the acknowledgment packets get back too late to reset the retransmission timer. These indications of false losses strongly impact on the overall performances of TCP in terms of reduced achievable throughput;
2) UDP has no notion of sequence numbers and out of order packets are forwarded to the application which, assuming it has a buffer (which is the case for all VoIP and most video applications), could reorder and utilise the packets;
3) DCCP's congestion control mechanism treats reordered packets as losses, although, similarly to UDP, the out of order packets are still delivered and may be of use to the application. However, the packets following the falsely detected congestion event will arrive at a lower rate (similarly to TCP). This follows the equation (1) driven sender rate and, depending on the level of reordering, could present a serious issue particularly for video applications. The artificially lowered rate could lead to losses between the sender side application and transport, which may not be able to handle the rate offered by the application.

In order to illustrate the impact of network reordering on DCCP/CCID3-CCID4, we drive a simple experiment where we simulate the network reordering effect by desequencing a number of selected packets with Linux NetEm [1]. In this experiment, we use a single bottleneck link of 1Mbit/s and uniformly distributed 5% random packets are delayed by a specified amount of time to create out of order packets (see NetEm parameters in the second column Table I). No losses (i.e. due to error link) are artificially introduced during the experiment. As shown in Table I, the higher the network reordering ratio, the higher the loss event rate (LER). As is to be expected based on equation (1), this increase leads to a decrease of the DCCP throughput (for comparison purpose the last column of the table gives the throughput obtained no network reodering is introduced).

Table I
DCCP BITRATE (X) FOR VARYING RTT AND REORDERING RATE

RTT	NetEm Reordering Parameters	X in Kbit/s	LER	X without network reodering
50ms	25ms +/- 8	935.3	1.66	961.3
100ms	50ms +/- 20	565.1	1.79	959.1
200ms	100ms +/- 40	500.2	1.83	954.5
300ms	150ms +/- 80	153.4	2.45	936.8
400ms	200ms +/- 160	90.6	3.11	941.2

The reduced performance of both transport and applications supports the validity of the idea of introducing a receiver side buffer specifically for handling out of order packets. However, a simplistic addition of a receiver buffer has a potential to increase the delay and jitter of the received data stream, therefore negatively impacting the applications QoE, as will be shown in Section IV. We aim to minimise the QoE degradation by optimising the buffer size to suit applications characteristics.

A. *Sizing the Reorder Buffer with Network Calculus*

We propose to implement a buffer to reorder packets before delivering them to the application. Therefore, we need to size this buffer as a function of the application constraints. Indeed, contrary to the elastic applications using the in-order delivery service provided by TCP, multimedia applications are characterized by strong delay bounds. For example, VoIP applications are characterized by a maximum "mouth-to-ear" threshold where a delay lower than around 200ms [13] is necessary to get a good communication quality, while video-conferencing is known to perform ideally when the delay does not exceed 100ms [5]. We thus propose to assess the size of this buffer following a given delay threshold provided by the application denoted D_{max}.

We denote $R(t)$ the rate at which data exits the IP level and enters the re-ordering buffer and $R^*(t)$ the output rate of the re-ordering buffer. When the re-order buffer is disabled: $R(t) = R^*(t)$.

In Fig. 1, L is corresponding to the maximum burst size and $R(t) = \lambda \cdot t + L$. We assume that D_{max} , is the maximum delay tolerated by the application and $d(t)$ be the delay induced by

[1]http://www.linuxfoundation.org/en/Net:Netem

the reordering queue ($d(t)$ might be considered as negligible as we only consider small buffer sizes). For the sake of simplicity, in Figure 1 $L = 1\ packet$ and $R(t) = 1\ pkt/s$.

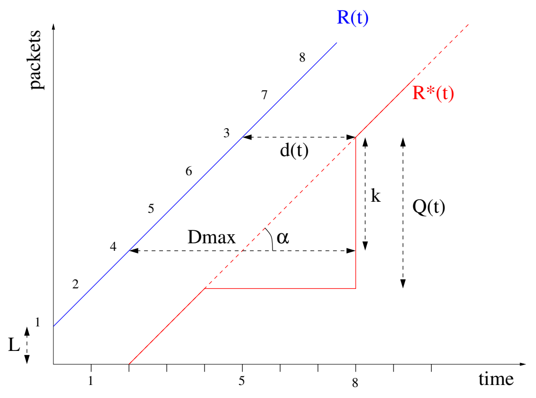

Figure 1. Arrivals curves $R(t)$ and $R^*(t)$

As shown, packets 1 and 2 arrive in-order and are transmitted to the application. Then, a rupture in the sequence numbers occurs when packet 4 arrives. Packet 4 and the following are enqueued in the reordering buffer until the reception of packet 3. When packet 3 is received, the reordering buffer is flushed without delay (this explains why $R^*(t)$ is perpendicular to the x-axis at $t = 8$).

Problem: knowing $R(t)$, what is the maximum value of the reordering buffer $Q(t)$ constrained by D_{max} ?

Following Fig. 1, we have $tan(\alpha) = \frac{k}{D_{max}-d(t)}$ and $Q(t) = k + L$, then:

$$\begin{aligned} Q(t) &= tan(\alpha) \cdot [D_{max} - d(t)] + L \\ &= \lambda \cdot \left[D_{max} - \frac{L}{\lambda} \right] + L \\ &= \lambda \cdot D_{max} \end{aligned}$$

As a matter of fact, λ is equal to X when DCCP-CCID3 is used, then $Q(t)$ becomes:

$$Q(t) = X \cdot D_{max} \tag{2}$$

In other words, for a given D_{max}, the maximum buffer size at instant t is given by the product of X and D_{max}. It also means that for a multimedia application characterized by a peak rate R_{peak}, we can size at the beginning of the connection $Q(t)$ to $R_{peak} \cdot D_{max}$.

B. Reorder Buffer for Unreliable Transport Protocols

Our proposal is in many ways similar to a flexible buffer common in VoIP and streaming video applications. It is applicable to DCCP or any other unreliable transport protocol which has the capability to detect the order of received data packets using e.g. sequence numbers.

Multimedia packets, on being received by the transport protocol receiving side, are, in the early stages when the initial applications data is being received, directed to a receiver buffer rather than passed on to the application. The buffer is filled up to the current maximum buffer size defined by equation (2) before the forwarding of data frames to the application begins.

The buffer is kept full at all times, and the received packets are accessed in regular intervals, as determined by the standard DCCP protocol. The buffer is scanned for the next packet, according to the required sequence number and forwarded from the buffer to the DCCP mechanism and the application.

Every incoming packet is assigned a relative position in the queue based on the maximum and minimum sequence number of existing packets in the buffer. Out of order packets based on their early or late arrival fill up the respective gaps in the queue. Dequeuing process simply selects a packet from the head of the queue which is the most relevant in order packet for the application. If an out of order packet which corresponds to the next-in-sequence packet is not available in the buffer at the time it is expected by the DCCP mechanism, it is considered lost and will contribute to the DCCP loss event rate.

III. Verification and Analysis

We have implemented our proposal within the DCCP transport protocol in the ns-2 simulator [11]. In all the simulations we use a simple dumbbell topology, with the DCCP sender and receiver connected with two routers which introduce reordering in the data packet stream. The reordering is achieved in the same way as described in the example shown in Table I. We vary the amount of delay on randomly selected packets, based on a uniform distribution, to create reordered packets. Using a uniform distribution is considered sufficient for the purpose of the initial evaluation, although we plan to consider other distributions in future work. We vary the applications data rate and RTT values and the percentage of reordered packets.

Our initial ns-2 implementation includes a fixed size buffer which we choose in the following way. For a video application, we assume the overall acceptable delay to be 100msec, consistent with [12] and a commonly used constant video rate of 1Mbit/sec. For the VoIP application, we choose two commonly used voice codecs, G.711 and G.729, with corresponding data rates (including IP and transport protocol overhead) or 80kbit/sec and 24kbit/sec. The QoE for VoIP applications [13] indicates the critical one way delay to be around 200ms, which if increased will more significantly impact the quality compared to delays below that value. As introducing the buffer will increase the already existing delay on the route between the sender and receiver, we allow for the buffer to contribute a D_{max} of 100msec to voice delay and 50msec to video delay. We note the most common delay on the Internet being around 50 msec [17]. Considering the average packet sizes of 500bytes for a H.264 video packet [5], 200 byes for G.711 and 60 bytes for G.729, we derive the maximum buffer size of 12 packets for video and 5 packets for VoIP.

A. *Simulation Results*

We perform a series of simulations to demonstrate the benefits of our proposal. We vary the reordering rate, RTT and buffer size and record the relevant DCCP parameters and the receiver side jitter. On the DCCP receiver side we also monitor the received rate and loss event rate and on the sender side we monitor the DCCP perceived RTT values. To enable thorough experiments, we have modified the ns-2 implementation [15] of the TCP test suite defined in [16] to handle DCCP with and without a reordering buffer. TCP specific parameters of course could not be monitored, however most of the generic parameters (e.g. RTT and throughput) have equivalent values for DCCP, although they may be derived in a different way.

To highlight the difference between the simulation results obtained using configurations with and without the reordering buffer, we define the efficiency ratio ε for a parameter z as follows:

$$\varepsilon(z) = \frac{z_b}{z_{nb} + z_b}$$

where z_b is the parameter value obtained by simulations which include the reordering buffer and z_{nb} is the corresponding parameter value resulting from simulations with no reordering buffer. For all experiments, when :

- $\varepsilon(z) = 0.5$: both schemes obtain the same performance;
- $\varepsilon(z) > 0.5$: the values obtained by the reordering buffer are higher than without reordering buffer;
- $\varepsilon(z) < 0.5$: the values obtained by the reordering buffer are lower than without reordering buffer.

We simulate an ideal application which will use the available data rate i.e. transmit the maximum amount of data using DCCP/CCID3 on a 1Mbit/sec bandwidth limited link. We present a subset of the derived efficiency ratio results for the same range of RTT and reorder buffer values, but choosing one reordering rate of 5% consistent with [7]. Figure 2 shows the resulting efficiency ratio for throughput, Figure 3 for the DCCP loss event rate and Figure 4 for the receiver side jitter.

The jitter, as shown Figure 4, obtained by both schemes is similar, i.e. the reordering buffer does not impact the jitter values in the received multimedia stream. We can observe that the losses are lower with the reordering buffer (see Figure 3). As a consequence, the resulting performance in terms of throughput is improved (see Figure 2). The throughput improvements range from small for a buffer size of 2 and low RTT (100-150msec) to a substantial 0.85 for a buffer size of 7. It is interesting to observe how the increased RTT reduces the reactivity of DCCP and amplifies the negative consequence of the reordered packets on the DCCP rate control mechanism. We note, as per the presented analysis of potential maximum buffer length values, that all the simulated buffer size values are lower than the estimated maximum buffer size of 12 packets for video. Additionally, that for a buffer size of 5 packets which has been considered the maximum for VoIP, we have an improvement (a throughput efficiency ratio of 0.6) even for the lowest simulated RTT value of 100msec.

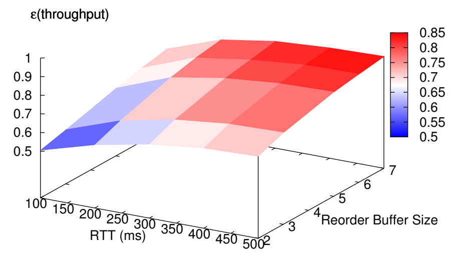

Figure 2. Throughput efficiency ratio

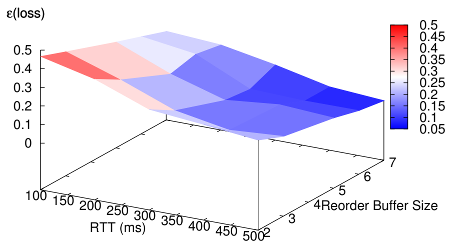

Figure 3. Loss event rate efficiency ratio

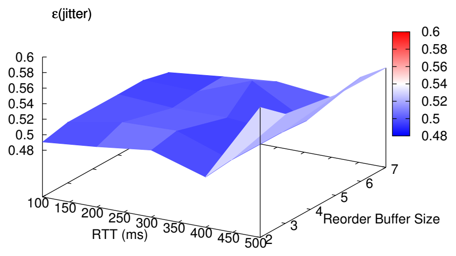

Figure 4. Jitter efficiency ratio

IV. Applications Quality Improvements

QoE for multimedia applications is most commonly represented by the value of the Mean Opinion Score (MOS). In changing network conditions, MOS is derived using an objective evaluation methodology, the E-Model, defined both for voice [13] and video [14] applications.

For voice applications, the E-Model's R factor quality metrics is computed based on the delay I_d and equipment impairments I_e, the latter being related to the specific voice codec's quality and ability to handle losses:

$$R = 94.2 - I_d - I_e$$

with:

$$I_d = 0.024d + 0.11(d - 177.3)U(d - 177.3)$$

where U is a unity function: $U(x) = 0$ if $x < 0$ and 1 otherwise.

Video quality metrics V_q is calculated using parameters which similarly relate to the video codec, however with a direct relationship defined for the frame and bit rates which can be varied, and to packet loss rate. We note that [14] only specifies QoE calculation for H.264 video codec being displayed on specific screen sizes.

$$V_q = 1 + I_{coding} exp\left(\frac{P_{plV}}{D_{PplV}}\right)$$

where I_{coding} represents the basic video quality related to the coding distortion under a combination of video bit rate and frame rate, the packet loss robustness factor D_{PplV} relates to codec robustness to packet loss and P_{plV} represents the loss rate. Streaming video quality only degrades with delay greater than 100 msec [12].

MOS values in the range of 5–1 respectively represent Excellent, Good, Fair, Poor and Bad quality. [13] and [14] define the corresponding R factor and V_q values.

We show example MOS values for VoIP and video applications using DCCP with and without the reordering buffer. We note that both voice and video applications may have a buffer to handle jitter and optionally reorder packets. To calculate MOS, we use simulation results for the DCCP rate available to the application and consider the additional delay introduced by either the DCCP or the application buffer. If out of order packets are not ordered by either of the buffers, they are considered lost by the application and included in MOS calculation.

Table II shows MOS for G.711 and G.729 voice codecs on a network with a 5% reorder rate, a 100 msec RTT and a buffer size of 5. It can be observed that buffering and reordering the data in either the transport protocol or the application has similar positive impact on the QoE of a voice conversation. We note that, due to low data rate of VoIP, the decreased DCCP rate is still high enough not to cause data losses.

Table II
MOS VALUES FOR G.711 AND G.729 VOICE CODECS

Options used	G.711	G.729
DCCP buffer	4.33	3.96
DCCP, no buffer; voice buffer	4.33	3.96
DCCP, no buffer; no voice buffer	3.87	3.27

Video applications have a higher data rate compared to VoIP and packet reordering resulting in a lower achievable DCCP rate may significantly impact the quality. Table III shows the MOS values for a H.264 codec video stream with a 5% reorder rate, 100 msec RTT and a buffer size of 12. We note the simulation results with the rate available to the video application being 898kbit/sec for our proposal and 314kbit/sec for standard DCCP. For a fair evaluation, we assume that the video application may have the capability to adjust the codec rate based on DCCP sender rate. Therefore, there are three cases: our proposal; standard DCCP with a rate adjustable video codec including an application buffer; and standard DCCP with a video buffer but no rate adjustment.

Table III
MOS VALUES FOR H.264 VIDEO CODEC

Options used	MOS 4.2" screen
DCCP-buffer	4.19
DCCP-no buffer; video buffer, rate adjustment	2.76
DCCP-no buffer; no video buffer, no rate adjustment	too low to calculate

It can be observed that the proposed transport protocol buffer has the potential to greatly increase the applications QoE and that, in the least favourable scenario when a low rate application is used, it will produce quality similar to the standard DCCP protocol. When used with video, it will significantly improve an otherwise unacceptably low quality.

V. Conclusion

We have analysed the performance degradation issues caused by packet reordering in unreliable transport protocols like DCCP and in multimedia applications. To combat these issues, we have proposed and evaluated the performance improvements achievable by a flexible receiver buffer implemented within a transport protocol. Our results show that this buffer allows to both increase the performance of the transport protocol and the QoE of the multimedia applications' user. We are currently considering to implement this scheme within the DCCP-TP [18] implementation.

VI. Acknowledgements

This research work has been supported by funding from National ICT Australia (NICTA). NICTA is a research organization funded by Australian Government research initiatives through Australian Research Council (ARC).

References

[1] Cisco Global mobile data traffic forecast update, Cisco, 2009

[2] E. Kohler, M. Handley, S. Floyd: Datagram Congestion Control Protocol (DCCP), RFC 4340.

[3] R. Stewart, Ed., Stream Control Transmission Protocol (SCTP), RFC 4960, September 2007.

[4] Jon C. R. Bennett, Craig Partridge, Nicholas Shectman: Packet reordering is not pathological network behavior. IEEE/ACM Transactions on Networking (1999).

[5] S. Wenger. H.264/AVC over IP. *IEEE Trans. Circuits Syst. Video Techn*, 13(7):645–656, 2003.

[6] Michal Przybylski, Bartosz Belter, Artur Binczewski: Shall we worry about Packet Reordering? TNC 2005.

[7] Sharad Jaiswal, Gianluca Iannaccone, Christophe Diot, James F. Kurose, Donald F. Towsley: Measurement and classification of out-of-sequence packets in a tier-1 IP backbone. IEEE/ACM Trans. Netw. (TON) 15(1):54-66 (2007)

[8] Ladan Gharai, Colin Perkins, Tom Lehman: Packet Reordering, High Speed Networks and Transport Protocol Performance. ICCCN 2004:73-78

[9] S. Floyd, E. Kohler, J. Padhye: Profile for Datagram Congestion Control Protocol (DCCP) Congestion Control ID 3: TCP-Friendly Rate Control (TFRC), RFC 4342.

[10] E. Kohler, S. Floyd: Profile for Datagram Congestion Control Protocol (DCCP) Congestion ID 4: TCP-Friendly Rate Control for Small Packets (TFRC-SP), Draft RFC 5622.

[11] The Network Simulator - ns-2, http://www.isi.edu/nsnam/ns/.

[12] S. Wenger: H.264/AVC Over IP, IEEE Trans. on Circuits and Systems for Video Technology, VOL. 13, NO.7, July 2003

[13] G.107: The E-model, a computational model for use in transmission planning, ITU-T, 2005

[14] G.1070: Opinion model for video-telephony applications, ITU-T, 2007

[15] TCP NS-2 stress testing script: http://www.hamilton.ie/net/tcptesting.zip
[16] Common TCP Evaluation Suite, draft-irtf-tmrg-tests-00.txt
[17] P. Pietzuch, J. Ledlie, and M. Seltzer: Supporting Network Coordinates on PlanetLab, In Proc. of WORLDS05, San. Francisco, CA, Dec. 2005
[18] DCCP-TP - DCCP Implementation Optimized for Portability, http://www.phelan-4.com/dccp-tp/tiki-index.php