Quantitative Analysis of the Effects Queuing has on a CCID3 Controlled DCCP Flow

Daniel Wilson
School of Information Technology
Murdoch University
Perth, W.A., Australia, 6150
Email: Daniel.Wilson@murdoch.edu.au

Terry Koziniec
School of Information Technology
Murdoch University
Perth, W.A., Australia, 6150
Email: Terry.Koziniec@murdoch.edu.au

Mike Dixon
School of Information Technology
Murdoch University
Perth, W.A., Australia, 6150
Email: Mike.Dixon@murdoch.edu.au

***Abstract*—While the Data Congestion Control Protocol (DCCP) shows much promise at becoming a protocol of choice for real-time applications in the future, there are relatively speaking, only a small number of academic papers purporting to its performance and its various nuances. This paper will describe the effects queuing and in particular queue sizes have on DCCP when CCID3 is selected as the congestion control mechanism. From results obtained through the experimentation described in this paper, a clear trade-off between packet loss rates and packet latency values was found to occur when different queue sizes were employed on the experimental network. It was found that employing small fixed sized queues on the network led to lower packet latencies but higher volumes of packet loss as a result of the queue size reaching its maximum threshold more frequently. Alternatively when large queue sizes were used, the number of packet loss events reduced significantly however, packet latency values increased. In addition to showing this impending trade-off empirically, this paper describes ways in which this phenomenon could potentially be exploited to allow DCCP to offer applications with a more tailored form of transportation protocol based on their particular needs.**

I. Introduction

When transferring data for real-time applications, the aim of the congestion control protocols is to provide the best performance in terms of both speed and reliability to the applications that will ultimately utilize the protocol. Although both are extremely desirable, performance in real-time applications is more dependent on timely delivery of data than reliable delivery of data. In this research, the Round Trip Time (RTT) metric is used to determine the timeliness of packet delivery. This value measures the total time taken for a packet to be sent and acknowledged across the network. Lower RTT values are indicative of faster delivery times of data packets. While, increases in RTT indicate that congestion is occurring along the network path and that there is a delay in delivering the data packet to the destination or along the return acknowledgement path. This experimentation will explore the effects prolonged RTT has on the performance of CCID3[1] and ways in which RTT caused by queuing of packets on transmit interfaces can be reduced in order to achieve the most desirable CCID3 transfer characteristics for real-time applications. In the next section, the scope of this paper will be discussed in the problem statement. This will be followed by a detailed description of the simulation toolkit and base DCCP model that were used to perform the experiments described in the experiment section. Results gathered from the experiment will then be presented followed by a discussion section. Following the discussion section, a number of suggestions as to how the performance of CCID3 controlled DCCP flows could be improved are presented. This paper concludes with a section outlining the experiment limitations followed by an overall conclusion section.

II. Problem Statement

One of the features of DCCP[2], is that it does not retransmit lost packets and therefore the effects of dropping packets are very different to those that would occur in TCP. In TCP dropped packets are retransmitted which reduces the effect the loss has on the application. Many protocols like those found in [3,4], have been designed and since proven to be effective in ensuring optimal performance through management of queue sizes. However most of these techniques are based on the premise that dropped packets will be retransmitted and therefore there is no real loss to the transfer other than added overhead and delays associated with the retransmission of lost packets. This notion is however less acceptable in DCCP as dropping packets for the purposes of rate control results in those packets being lost permanently.

Not retransmitting lost packets would suggest that DCCP should be treated more in line with UDP which does not retransmit lost packets. However, when a loss occurs in UDP transfers, the rate is not reduced as a result of the loss event. Instead, in UDP the transmit rate is largely governed by the sending application which is in most situations oblivious to the loss. While this approach has worked fairly successfully in the past, as applications become more network resource intensive, network congestion resulting from no congestion control in UDP has been cited as being unsustainable[2].

Although DCCP is much simpler to understand conceptually than TCP via the alleviation of retransmission and the compounding effects that retransmission can have on a congested network, the inimitable effects of not retransmitting (like UDP) and slowing the transmission rate down simultaneously (like TCP) have not to date been fully explored. These experiments seeks to quantify the effects queue size has on CCID3 performance over a single flow. By doing this the effects of rate

978-1-4577-2059-8/11/$26.00 ©2011 IEEE

control without lost packet retransmission will be showcased. Although these experiments are relatively simplistic, the experimentation aims to procure a set of baseline performance results which it is hoped future research can reference and draw inferences from in relation to this particular facet of DCCP behaviour.

III. The queue size coefficient

If CCID3 passes information down to the data link layer at a rate that is faster than the rate at which the interface on the sending device is capable of sending, then the packets will either need to be dropped or stored in a queue. How quickly the interface is able to service these packets is dependent on the link speed and serialization time required to transmit those packets. In some cases the bandwidth available on the transmission interface may exceed the speed needed to transmit packets passed down from the transport layer and therefore no queuing of packets would be required. In these circumstances the actual transmit rate is governed by the CCID3 transmit rate. This experimentation will only discuss scenarios where the CCID3 transmit rate exceeds the interface transmission speed and queuing or dropping of packets on the transmitting interface is required. In addition this section will only discuss a single DCCP flow. In reality there would likely be multiple simultaneous flows utilizing the medium which would add to the size of an outgoing queue. Ways in which multiple flows can be handled will be discussed in the discussion section of this experiment. While this experimentation is performed solely on the transmit interface on the sending device, the effects queuing has on DCCP transmissions will apply on all transmit interfaces located on intermediate devices between the sender and the receiver.

IV. Modelling of the standard for Experimentation

Modelling in this research was carried out using the Opnet Modeler simulation tool-kit [5]. The base code used to implement the standard DCCP and CCID3 protocols in Opnet was developed by Xiaoyuan in [6]. Minor modifications were made to the model found in [6] in order to bring the code into alignment with RFC5762 using parameters and recommendations found in [7]. At the time of writing this paper, there is no ratified IETF standard for DCCP and therefore the most specific specifications for implementing DCCP and CCID3 are those found in the relevant RFCs mentioned in sections above.

For all modelling contained herein, the baseline model created by [6] was left largely unchanged as the source code used appears to be similar if not identical to the current Linux and NS2 versions of the DCCP code. This code has become the de facto code used when modelling DCCP in the research community and hence the decision to make use of it. As the work in [7] which essentially maps RTP to DCCP, was not yet written at the time when the base model described in [6] was created, changes were made to ensure the code was compliant with this proposed RFC. The majority of changes to the base model in [6] occurred at the application layer. These changes were done to ensure a correctly sized simulated RTP packet stream was created to utilize DCCP and to ensure this was in-line with the proposed standard presented in [7].

V. Experiment Design and Methodology

To carry out the experimentation needed to determine the effect the queue size has on the performance of CCID3, four (4) simple topologies were created. In each topology a single sender and receiver were connected to one another via two set speed links. Each of the links was only used in one direction to ensure there was no inadvertent contention for bandwidth. Each of the 4 topologies employed a unique link speed on the link between the sender and the receiver ranging from a very slow 96kbps (topology A) through to a faster 5096kbps (topology D). Once the topologies were created, multiple iterations of the simulation were then run on each topology. Each of these iterations used a different maximum queue size limit on the outgoing transmit interface on the sender node. In doing this, the queue size was never allowed to exceed the predetermined amount specified for the iteration. When the queue filled to capacity, new packets arriving were simply dropped to ensure the queue never exceeded the set amount. The queue size limits were set at 10, 20, 50, 100 and 200 packets.

This experimentation focuses on the first 120 seconds of the session between the sender and the receiver. In addition a complete set of result samples were also taken at 50 seconds to determine the extent to which the slow-start phase found in CCID3 impacted on the results. Additionally, it should be noted that while this experimentation will relate specifically to the queue located on the transmitting interface on the sender, the results are applicable to any queue located between the sender and the receiver. It is therefore plausible that the results herein can therefore be applied to the queues on intermediate devices such as Internet routers and switches that employ queues.

VI. Results

A. Round trip times

When the transmit interface's link speed is not able to service packets passed down from the higher layers quickly enough then queuing on the outgoing interface occurs.($IfRecv\_rate > Int\_tx\_rate = increase\_in\_queue\_size$). When the queue size begins to grow, packets that are appended to the queue are subjected to the delay time needed to send all the packets placed into the queue prior to them. As the packets take longer to arrive at their destination, there is an increase in the RTT value which has an impact upon CCID3s performance and ultimately the performance of the application. The minimum time which a packet will be subjected to in a queue can be expressed as follows assuming that the queue is operating on a first-in-first-out basis (FIFO) and that the outgoing interface link is able to be utilized at full capacity.

Due to the construction of the application layer in this experiment which was configured to send packets to the DCCP

$$MinExpectedDelay = \frac{(Preceeding\_no\_pkts\_in\_queue)*8}{Interface\_tx\_rate(kilobits/sec)}$$

Fig. 1. Minimum expected queue time calculation in a FIFO queue if no packet purging occurs.

layer as quickly as the DCCP layer was capable of processing them, the results demonstrated below show the various Round Trip Times (RTT) that can be expected when different queue sizes are employed in environments offering ideal network conditions i.e., no loss or contention. In the figure shown below (Fig.2), the results gathered from the experimentation for RTT values extracted from each of the four topologies are presented.

		RTT values for initial 120 seconds			
Bandwidth	**Queue size**	**Max**	**Min**	**Avg**	**Std Dev**
96kbps	10	0.2039	0.070200	0.189082	0.027819
	20	0.3775	0.070200	0.299734	0.092333
	50	0.757	0.070200	0.421273	0.189637
	100	1.108	0.070200	0.464929	0.201554
	200	2.233	0.070200	0.567654	0.371687
512Kbps	10	0.0908	0.017800	0.036290	0.002298
	20	0.0908	0.024300	0.066426	0.006239
	50	0.1691	0.029500	0.136037	0.041222
	100	0.3305	0.040100	0.212445	0.089467
	200	0.4167	0.040100	0.191859	0.084678
2048Kbps	10	0.0902	0.005800	0.007970	0.001683
	20	0.0902	0.006400	0.016065	0.001924
	50	0.0902	0.009700	0.038448	0.004590
	100	0.0902	0.010200	0.066811	0.019863
	200	0.1641	0.014800	0.090598	0.044027
5096kbps	10	0.09	0.020000	0.002815	0.001123
	20	0.09	0.003300	0.005929	0.001411
	50	0.09	0.004000	0.015174	0.002045
	100	0.09	0.005500	0.029170	0.029170
	200	0.09	0.006000	0.048873	0.018290

Fig. 2. Results of Round Trip (RTT) times experienced during experimentation.

From the results, it is clear that employing smaller queue sizes results in significantly smaller RTT values as a result of the queuing process. The smallest queue size of 10 resulted in the smallest RTT values in all four test topologies. As the queue size increased so too did the average RTT for the transmission. Statistical analysis supports this hypothesis as there is clear correlation between queue size and the average RTT for the transfer. An increase in queue size directly correlates to the increase in added end-to-end delay and increase in RTT value (supported at the 99.995% confidence level).

Reasoning as to why the RTT value is much smaller when a small queue size is employed is due to the fact that in small queues the compounding effect of the delay needed to service preceding packets in the queue is smaller. When the queue size is allowed to grow larger, so too does the delay packets are subjected to by the process of servicing the preceding packets in the queue. In a queue of size 10, there are only ever a maximum of 9 packets that need to be serviced prior to the servicing a new packet whereas in a larger queue of say 200, there are 199 packets which may need to be serviced prior to a new packet being able to be sent.

B. Packet Loss

Before discussing packet loss and presenting results obtained from the experimentation, one very important thing to note is that packet loss occurs differently depending on which stage or phase the congestion algorithm is in. The quantity of packets lost in slow-start may not necessarily be indicative of the packet loss ratio that will occur during the normalized equation based rate control phase of CCID3. For this reason, results of packet loss in this experimentation are recorded at two stages during the 120 second test session. The first set of results is extracted at 50 seconds (where slow phase has ended in all experiments) and the second set at the end of the 120 seconds experiment session.

Fig. 3 below shows the results relating to the packet loss experienced when different queue sizes and network link speeds are used after 50 seconds and 120 seconds of the DCCP CCID3 flow.

	50 seconds					120 seconds			
	Queue Size↓	Pkts sent	Pkts Recv	Pkts lost	Pkt Loss %	Pkts sent	Pkts Recv	Pkts Lost	Pkt Loss %
96kbps	10	2528	2511	17	0.6725	6529	6492	37	0.5667
	20	2496	2484	12	0.4808	6484	6465	19	0.2930
	50	2477	2454	23	0.9285	6458	6435	23	0.3561
	100	2434	2398	36	1.4790	6415	6379	36	0.5612
	200	2339	2264	75	3.2065	6320	6245	75	1.1867
512Kbps	10	13457	13381	76	0.5648	34805	34613	192	0.5516
	20	13446	13417	29	0.2157	34713	34649	64	0.1844
	50	13392	13369	23	0.1717	34631	34601	30	0.0866
	100	13424	13391	33	0.2458	34658	34623	35	0.1010
	200	13257	13185	72	0.5431	34489	34417	72	0.2088
2048Kbps	10	53855	53482	373	0.6926	139368	138411	957	0.6867
	20	53624	53520	104	0.1939	138710	138449	261	0.1882
	50	53603	53573	30	0.0560	138560	138502	58	0.0419
	100	53553	53511	42	0.0784	138487	138439	48	0.0347
	200	53483	53397	86	0.1608	138413	138326	87	0.0629
5096kbps	10	134000	132813	1187	0.8858	347170	344140	3030	0.8728
	20	133188	132867	321	0.2410	344975	344194	781	0.2264
	50	133145	133090	55	0.0413	344543	344417	126	0.0366
	100	133158	133118	40	0.0300	344504	344445	59	0.0171
	200	133096	133022	74	0.0556	344427	344349	78	0.0226

(Corrected to 4 decimal places)

Fig. 3. Packet loss rates experienced during experimentation.

From the results, it becomes apparent that small queues (10 packets)and large queues (200 packets) result in the largest number of packet drops compared to queue sizes of 20, 50 and 100 at the tested speeds. Deeper analysis however shows that once the slow phase has ended, the larger queue size (200) results in the smallest number of packet drops in the period between 50 seconds and 120 seconds. The large majority of packet drops that occur in the topology employing queue sizes of 200 occur due to the slow-start mechanism and large number of packet drops that result in the eventual termination of the slow-start phase in large queues. Once this phase has completed, the number of dropped packets becomes negligible when compared to the smaller queues tested as drops occur at much more infrequent intervals.

In bigger queues the number of packet drops does not increase as much after the initial slow-start phase. Once this phase is over, there is very little packet drop. This is because there are no drops due to the queue reaching its maximum size when the transfer rate stabilizes. In addition the average RTT

time is much higher and therefore the algorithm is less reactant to packet loss. This is due to there being a much longer period between acknowledgements of packets compared to smaller queues where the average RTT is much smaller and the congestion algorithm more reactant.

On the other hand the smaller queues experience continual packet drops throughout the life of the session as the maximum queue size is reached continually throughout the transfer. The results show statistically that the smaller the queue size is, the more sustained the packet drop rate will be for the duration of the session even after the initial slow-start phase has ended. Support for this hypothesis is shown in the 10 and 20 queue size topologies where there is a continual increase in the number of packets lost/dropped in the 50 second to 120 second phase of the transfer. An increase in the queue size has a directly inverse correlation to the number of packet drops that occur in the period after the initial 50 seconds.

The abnormal results found in the 96kbps topology where the queue size of 200 was found to be produce the highest number packet drops was found to caused as a result of the experiment duration (120 sec) and the very slow link speed used for testing. At 96kbps, the network was not able to refill transmission queue to above 40 packets within the allotted 120 seconds after the initial slow-start phase ended. Although not shown in the results table, this experiment duration was later increased to 10 minutes and the results obtained showed that a packet queue size of 200 resulted in the lowest number of packet drops overall when compared to the other queue sizes after a 10 minute transfer period. The queue size of 200 resulted in a total of 82 packets losses after 10 minutes compared to the 216 packets lost for the queue size of 10 over the same period when 96kbps links were employed between sender and receiver.

C. Throughput and Goodput

From these results, there is clear evidence to suggest that larger queue sizes result in fewer packet drops over sustained periods. As also shown above however, employing larger queues to reduce packet loss comes at the expense of increased delay being added to the packets RTT value. Clearly there is a need for the application to determine which of these characteristics is more important to it. In an attempt to add further clarity to this trade-off, an additional group of characteristics relating to the transfer will now be presented.

Throughput as described herein describes *the rate of the total number of packets sent per second (not necessarily received)*. The term goodput will be used to describe *the number of successfully received packets sent per second*. Finally the goodput ratio is used to describe *the number of successfully received packets as a fraction of the total number of packets sent*.

In the figure below (Fig.4), the results pertaining to throughput, goodput and goodput ratio (percentage of loss in Fig.4) experienced after 50 seconds and 120 seconds of the DCCP CCID3 flow are shown.

		50 Seconds			*120 Seconds*		
	Queue size ↓	**Throughput (Pkts/Sec)**	**Goodput (Pkts/Sec)**	**Loss %**	**Throughput (Pkts/Sec)**	**Goodput (Pkts/Sec)**	**Loss %**
96kbps	**10**	50.56	50.22	0.6725	54.4083	54.1000	0.5667
	20	49.92	49.68	0.4808	54.0333	53.8750	0.2930
	50	49.54	49.08	0.9285	53.8167	53.6250	0.3561
	100	48.68	47.96	1.4790	53.4583	53.1583	0.5612
	200	46.78	45.28	3.2065	52.6667	52.0417	1.1867
512Kbps	**10**	269.14	267.62	0.5648	290.0417	288.4417	0.5516
	20	268.92	268.34	0.2157	289.2750	288.7417	0.1844
	50	267.84	267.38	0.1717	288.5917	288.3417	0.0866
	100	268.48	267.82	0.2458	288.8167	288.5250	0.1010
	200	265.14	263.7	0.5431	287.4083	286.8083	0.2088
2048Kbps	**10**	1077.1	1069.64	0.6926	1161.4000	1153.4250	0.6867
	20	1072.48	1070.4	0.1939	1155.9167	1153.7417	0.1882
	50	1072.06	1071.46	0.0560	1154.6667	1154.1833	0.0419
	100	1071.06	1070.22	0.0784	1154.0583	1153.6583	0.0347
	200	1069.66	1067.94	0.1608	1153.4417	1152.7167	0.0629
5096kbps	**10**	2680	2656.26	0.8858	2893.0833	2867.8333	0.8728
	20	2663.76	2657.34	0.2410	2874.7917	2868.2833	0.2264
	50	2662.9	2661.8	0.0413	2871.1917	2870.1417	0.0366
	100	2663.16	2662.36	0.0300	2870.8667	2870.3750	0.0171
	200	2661.92	2660.44	0.0556	2870.2250	2869.5750	0.0226

(Corrected to 4 decimal places)

Fig. 4. Throughput, Goodput and percentage of loss experienced after 50 seconds and 120 seconds.

By expressing the results in terms of throughput, goodput and goodput ratios, the trade-off between having faster transfers and more reliable transfers as governed by the queue size once again becomes apparent. In all cases where the smallest queue size is used (10 packets), the best throughput rate is achieved. However, this rate does not factor into account the number of lost packets that occur as a result of the maximum queue size being reached more frequently.

In order to have transfers with the best loss ratio (i.e., the percentage of lost packets) the results suggest larger queue sizes should be employed. Once again, the results show that depending on whether the congestion control algorithm is in slow-start or equation based flow control mode will have a profound impact on the number of lost packets with the larger queues losing more packets during this phase. Once the slow-start phase has ended, there is a clear increase in throughput, goodput and loss ratio percentage for the larger queues when compared to the smaller queues after the transition into equation based flow control mode. The increase in these values post slow-start phase correlates directly with the queue size, with larger queue sizes demonstrating larger increases in goodput. Because the number of packet losses occurring in the small queue topology is more frequent, the increase in throughput, goodput and loss ratio percentages is lower when compared to the other queue sizes employed.

The results do suggest that a moderate queue size of 50 packets results in the best overall packet loss ratio for the 120 second period in all the experiment topologies except where a link speed of 2048kbps was employed. In this case the slightly larger 100 sized packet queue was the best performer.

VII. Discussion

The results shown above demonstrate that through the utilization of smaller queue sizes, optimal RTT values are achieved. These however come at the expense of higher rates of packet loss. What effect these lost packets would have on

the application was not studied given the wide multitude of applications that require real-time data transfer. However, they are expected to be quite profound as there is no retransmission in DCCP. If the application is able to withstand the sustained packet loss at the rates shown above and seeks very small RTT times then small queue sizes should definitely be used. A prime example of this would be a stock ticker where data that is *as close to real-time as possible* is more desirable than delayed more reliable data flows. As such the higher loss is more desirable than delayed data.

On the contrary however, there are real-time applications that require *"as close to real-time as possible"* transfers but are less resistant to loss. Examples of these include voice and live video transfers which require low levels of RTT but also are severely impacted by loss. Given the results, four possible approaches to situations where high loss is less acceptable to the application will now be proposed.

A. Empowering the application approach

If the application layer were to be made responsible for the retransmission of lost packets then an accurate determination as to whether or not a particular packet should be retransmitted would provide more efficient data transfers. For example in a Video transmission (H.264[8])the sender may determine that a lost packet containing a B-Frame need not be retransmitted but a packet containing an I-Frame should be. If the application does this, then queue sizes could be made very small to ensure fastest transfer speeds. As retransmission is occurring, the extra loss inflicted through the utilization of a smaller queue is somewhat reduced. This scheme is only possible if the RTT is small enough to allow the retransmission to occur within the useful lifespan of the data.

B. The happy-medium approach

Another solution that can be used by the protocol is to select a *happy-medium* approach whereby a medium (relative) sized queue is selected on the outgoing interface that offers neither the best nor the worst performance in terms of packet loss ratios or RTT. In the experimentation, it was shown that queue sizes of 20, 50 and 100 offered relatively low levels of packet loss in addition to relatively low RTT values. In situations where loss is less acceptable, but RTT must be kept below a certain threshold, this solution would be feasible if the level of packet loss was less than the application's loss threshold and the RTT value was less than acceptable delay. One of the limitations with this approach is that this scheme is not ideal where multiple streams with different requirements are operating on the link simultaneously.

When more than one stream exists, packets from the various streams are placed together into a single outgoing transmit queue in a manner determined by the transport layer protocol. When placing packets from multiple streams into a single outgoing queue it quickly becomes apparent that an approach where the applications are made responsible for setting a maximum queue sizes on outgoing transmit interfaces becomes problematic. Where multiple streams co-exist a *referee* algorithm would be required to determine what the queue size should be to balance the needs of the various applications. As applications may vary greatly in their requirements the ability for such a refereeing algorithm to find a queue size suitable for two or more applications with disjunct requirements would be extremely difficult and computationally intensive.

C. Categorization and prioritization of Packets

While the applications clearly cannot be left to choose the size of the outgoing queue size, this does not however discount the value of the findings above which show that application performance can be optimized through careful manipulation of queue sizes. In future research a queue categorization scheme will be introduced whereby packets are placed into unique categories depending on their application's defined requirements and the amount of useful lifespan the packets possess. Once categorized, packets will then be scheduled into the queue in a non-linear manner ensuring they are transmitted in the most efficient manner possible.

Such a scheme removes the need for the application to manipulate the queue sizes and alleviates the possibility of conflict between competing application streams. While this scheme would render the *"happy-medium"* approach null and void, it is predicted that a categorization scheme would integrate well with first approach whereby the onus is placed on the applications for them to be responsible for packet retransmissions. By categorizing packets arriving from various application streams, the protocol is able to maintain a single queue but offer the best performance to all applications utilizing an interface through various scheduling techniques.

D. Speeding up the termination of the slow-start phase

The results indicate that during slow-start, relatively speaking, a very large number of packet drop events occur. As some real-time applications may only have short durations, such as voice calls, the results suggest that there may be some merit in prematurely causing an end to the slow-start phase to reduce the number of packet drops occurring. This would likely come in the form of an updated slow-start anti-oscillation mechanism that is designed specifically for short lived real-time applications. Future research is recommended to determine the effects and early exiting of the slow-start mechanism would have of CCID3 performance.

VIII. Limitations of this experimentation

The simplicity of this experiment was deliberate in order to validate the importance the queue size and increased RTT values can have on a single DCCP stream. In addition two other objectives were also met through this experimentation. First the experiments aimed to showcase and validate that the simulation model of the DCCP protocol was working correctly and in accordance with the various RFC Standards. Secondly to generate a baseline of the results that DCCP is capable of producing in ideal situations. While these objectives have been met, the experimentation does have a number of limitations which will now be discussed.

A. Application Send Rate

The RTP layer application was designed in such a way in this experimentation that it sent packets to the DCCP layer as rapidly as the DCCP layer was able to process them. This was done to ensure that there was complete utilization of the network link in the topology. In the real world applications would not act in such a manner. Instead a steady stream of packets would be passed to the DCCP layer at a rate governed by the application's requirements and not by the bandwidth available at the physical layer (with the exception of where the application send rate was greater than the available bandwidth could service the packets at). To illustrate this, an application making use of the G7.11[9] codec would send a constant stream of 64 Kilobits/second across any links capable of servicing this rate and provided there was no other contention for the bandwidth irrespective of the links speed. This would equate to approximately 50 packets per second given the standard sampling period for G7.11. For this reason this experiment does not possess an application layer that is truly indicative of what would be found in a typical real world network. This research should therefore be framed in the context where it is only applicable to environments where congestion is being experienced on networks and where queuing is taking place due to a disparity in received packets rates and sent packet rates. The rapid increase in number of emerging real-time applications and their increased need for higher amounts of bandwidth would suggest this scenario is going to become increasingly common.

B. Effects of Packet drops on the application

In addition to the unrealistic application send rate, this experimentation does not explore the impact packets loss has on the application. The reason for not entering this area of research is simple. There is such a diverse range of real-time applications emerging with different requirements that it would be nearly impossible to comprehensively cover the effects packet loss would have on each of them. This becomes a very application specific task and for this exact reason the approach introduced above whereby the application was made responsible for its own retransmissions would be ideal as it would allow better decisions to be made relating to the importance of the lost packet. Packets which have high value to the application could be retransmitted by the application whereas lost packets with little or no importance could simply be ignored. DCCP will not retransmit the lost packets so it is expected applications adopting the protocol will eventually move in this direction of self-governed retransmission.

C. Multiple Stream Interaction

As already discussed, this experiment focuses on a single flow. In the real world multiple simultaneous flows would likely occur throughout the network. The results presented here are therefore the best case results a single CCID3 stream in ideal network conditions could expect. The interaction and effects of competing streams vying for the same bandwidth have not been explored. It is hoped in future research these phenomena will be explored further. In addition the network does not take into account normal incidences that would commonly occur in real-world networks such as unexpected packet drops, fluctuations in delay and congestion or the use different route paths for a single flow.

IX. Conclusion

The results have shown that employing small transmit queues lessens the RTT value but leads to a higher and more sustained packet loss rate. Unlike TCP which retransmits lost packets, these lost packets are not sent again in DCCP which could potentially have a negative impact on the application. It is concluded that in order to achieve optimal transfers via DCCP and CCID3, the application layer should be made responsible for retransmission of lost packets and potentially influence the queue size for a particular flow. In doing so the application designer could then tailor the transmission characteristics to suit their need for either minimal packet loss or minimal RTT values (delay). In addition, the need for a categorization scheme is alluded to that is able to take control of queue depths for multiple simultaneous streams from different applications. How such a scheme would work will be presented in subsequent papers. In addition to demonstrating the effects queuing and RTT has on CCID3 performance, this paper has also served to introduce and verify that the simulated implementation of DCCP, CCID3 and RTP are functioning correctly and operating in accordance with the various RFCs. Finally through the experimentation listed above, this paper has also produced a collection of baseline DCCP performance data to which later comparisons will be made as new contributions are added to the various DCCP standards.

References

[1] Handley, M., et al., TCP Friendly Rate Control (TFRC): Protocol specification. Internet request for Comment, 2003. RFC 3448.

[2] Kohler, H. and S. Floyd, Datagram Congestion Control Protocol Internet Request for Comment, 2006. RFC 4340.

[3] Floyd, S. and V. Jacobson, Random early detection gateways for congestion avoidance. IEEE/ACM Transactions on networking, 1993. 1(4): p. 397-413.

[4] Feng, W., et al., The BLUE active queue management algorithms. IEEE/ACM Transactions on Networking (TON), 2002. 10(4): p. 513-528.

[5] Opnet-Technologies, Opnet Modeler, in Available: $http : //www.opnet.com/solutions/network_rd/modeler.html.2007 - 2011$. Accessed: 23.01.2011

[6] Xiaoyuan, G., TU Braunschweig DCCP Model. 2006, Opnet User Contributed Model Repository.

[7] Perkins, C., RTP and the Datagram Congestion Control Protocol (DCCP). Work in Progress, 2007.

[8] Wiegand, T., et al., Overview of the H. 264/AVC video coding standard. IEEE Transactions on circuits and systems for video technology, 2003. 13(7): p. 560-576.

[9] Rec, I., G. 711. General Aspects of Digital Transmission Systems Terminal Equipments-Pulse Code Modulation (PCM) of Voice Frequencies, 1972.