Incorporating Deadline Scheduling
Into DCCP

This thesis is presented for the degree of Doctor of Philosophy of Murdoch University

By Daniel Edward Wilson (BSc with Honors)

Murdoch University
2012
Originality Statement

I declare that this thesis is my own account of my research and contains as its main content work which has not previously been submitted for a degree at any tertiary education institution.

Signed: ____________________________________________

(Daniel Edward Wilson)
Abstract

In current TCP, UDP and DCCP networks, the finite nature of real time data is not taken into consideration when scheduling of packets occurs on intermediate devices. To overcome this, the research presented in this thesis incorporates deadline scheduling into DCCP. This allows packet age information to be transferred by the DCCP protocol which provides intermediate devices with the ability to make more informed transportation decisions for real time data. By embedding two variables, in the form of options into DCCP-Data packets, a number of new prioritization and scheduling mechanisms that utilize packet age information are made possible. This thesis will show how deadline scheduling is incorporated into DCCP in a stable, backward compatible and DCCP standard compliant manner.

Once deadline scheduling is incorporated into DCCP, the focus of the thesis then shifts towards mechanisms that can be implemented on intermediate devices that make use of the packet life information. To begin, five unique packet discard mechanisms that purge stale packets from the network are presented. The purpose of these mechanisms is to remove stale packets from the network using intermediate devices in the network in order to free network resources for non stale packets utilizing the same infrastructure. Experimentation carried out to investigate the efficiency and benefits to DCCP performance offered by each of these five mechanisms is shown. The experimentation also explores fairness amongst competing flows when the mechanisms are activated.

Following this, a novel probabilistic scheduling (PBS) mechanism is introduced that predicts the probability a packet has of arriving at its intended destination network within its useful lifespan. Once this probability is calculated scheduling decisions are then made based on this value in order to offer optimized delivery to real time data. In order to calculate this probability, the PBS mechanism utilizes metrics from
the Cisco EIGRP routing protocol. Experiments carried out using the PBS mechanism demonstrate that the PBS mechanism improves DCCP performance in networks where high levels of stale packets occur.

Overall, this research presents a new approach to the transportation of real time data in DCCP networks and aims to improve DCCP adoption through the improved performance capability added to the protocol by this research.
## Contents

1 Introduction 1
   1.1 Background .................................................. 1
   1.2 Research problem ........................................... 3
   1.3 Purpose of this research ................................... 3
   1.4 Research approach .......................................... 5
   1.5 Organization of the thesis .................................. 6
   1.6 Publications ................................................. 8

2 Literature Review 9
   2.1 Introduction .................................................. 9
   2.2 Deadline based network resource management .................. 10
   2.3 Packet life discard schemes .................................. 12
   2.4 Discussion of deadline scheduling in DCCP .................... 15
   2.5 Overview of the DCCP standards ............................... 16
      2.5.1 DCCP connection establishment, management and termination
      2.5.2 Packet sequencing and numbering in DCCP ................. 18
      2.5.3 DCCP packet format ....................................... 18
      2.5.4 Modularity in DCCP ....................................... 19
      2.5.5 Feature negotiation options ............................... 20
      2.5.6 Congestion control in DCCP ............................... 20
   2.6 Anatomy of a typical CCID3 controlled session ............... 24
      2.6.1 The function of CCID3 in DCCP ........................... 24
      2.6.2 Phases of a CCID3 flow .................................. 25
      2.6.3 Slow start phase .......................................... 26
      2.6.3.1 Preventing oscillations ............................... 28
   2.7 Demonstration of transmit rate calculations during slow start
   2.8 Conclusion .................................................... 36

3 The effect of queue sizes on CCID3 performance 37
   3.1 Introduction .................................................. 37
   3.2 Background ................................................... 38
   3.3 Effects of queuing on CCID3 controlled DCCP flows .......... 38
   3.4 Experiment aims .............................................. 40
   3.5 Experiment design and methodology ........................... 40
   3.6 Results ....................................................... 41
      3.6.1 Round trip times (RTT) .................................. 41
      3.6.2 Packet loss ............................................... 43
3.6.3 Throughput and goodput ........................................ 45
3.7 Discussion .......................................................... 47
  3.7.1 Various forms of real time applications ....................... 47
3.8 Limitations of this experimentation ................................. 50
  3.8.1 Application send rate ............................................ 50
  3.8.2 Effects of packet drops on the application .................... 51
  3.8.3 Multiple stream interaction ..................................... 52
3.9 Conclusion ........................................................... 52

4 Incorporating deadline scheduling into DCCP .......................... 54
  4.1 Introduction .......................................................... 54
  4.2 Motivation and background ........................................ 55
    4.2.1 Deadline scheduling in DCCP .................................. 57
    4.2.2 How the deadline scheduling mechanism works ............... 59
      4.2.2.1 High level overview ....................................... 59
      4.2.2.2 NTP synchronization ....................................... 61
      4.2.2.3 DCCP header modification ................................ 62
      4.2.2.4 Option fields ............................................. 63
    4.2.3 Non compliant devices ......................................... 65
    4.2.4 Possible uses of the newly proposed option fields ......... 66
  4.3 Experimentation ..................................................... 66
    4.3.1 Simulation toolkit .............................................. 67
      4.3.1.1 Timing and NTP ............................................ 67
      4.3.1.2 Application layer design for simulated model ........... 68
    4.3.2 Network topology .............................................. 69
  4.4 Experiments testing the effects of incorporating deadline scheduling into DCCP Networks ......................... 72
    4.4.1 Experiment aims ................................................ 72
    4.4.2 Methods of results collection ................................ 72
    4.4.3 Experiment 1: Detriment to window growth rates ............ 73
      4.4.3.1 Results of experimentation ................................ 73
      4.4.3.2 Window growth rates in highly congested networks .... 76
      4.4.3.3 Discussion of observed window growth times ............ 79
    4.4.4 Experiment 2: Throughput, goodput and RTT ................. 80
    4.4.5 Experiment 3: Effects of deadline scheduling on packet loss ........................................ 87
      4.4.5.1 Reaction to packet loss events during stabilized CA .... 87
      4.4.5.2 Reaction to packet loss over a sustained period ......... 90
    4.4.6 Limitations of experiments ................................... 94
      4.4.6.1 Queue size ................................................ 94
      4.4.6.2 Determining stabilization points ........................ 94
    4.4.7 Conclusion of experiments ..................................... 95
  4.5 Chapter conclusion ................................................ 95

5 Lifetime Discard Protocols for DCCP .................................. 97
  5.1 Introduction ........................................................ 97
  5.2 Background ........................................................ 98
  5.3 Scope of the problem .............................................. 98
  5.4 Overview of the packet life discard mechanisms ................. 100
    5.4.1 Deletion method for purging stale packets ................. 101
      5.4.1.1 Advantages of the scheme ................................ 102
      5.4.1.2 Disadvantages of the scheme ............................. 102
      5.4.1.3 Additional variations explored on the basic delete mechanism ........................................ 104
    5.4.2 ECN used as a stale packet control mechanism ............. 106
    5.4.3 Phantom packet method for purging stale packets .......... 108
## List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Example of DCCP sequence and acknowledgment numbering</td>
<td>18</td>
</tr>
<tr>
<td>2.2</td>
<td>Components in a typical DCCP packet</td>
<td>19</td>
</tr>
<tr>
<td>2.3</td>
<td>Typical DCCP behavior. Transmission rate</td>
<td>25</td>
</tr>
<tr>
<td>2.4</td>
<td>Enlarged slow start phase of DCCP transmission</td>
<td>27</td>
</tr>
<tr>
<td>2.5</td>
<td>Transmission rate, round trip time (RTT) and queue depth during slow start</td>
<td>30</td>
</tr>
<tr>
<td>2.6</td>
<td>Transition from slow start phase to equation based flow control</td>
<td>32</td>
</tr>
<tr>
<td>4.1</td>
<td>Generic DCCP header</td>
<td>63</td>
</tr>
<tr>
<td>4.2</td>
<td>Additional NTP option field for deadline scheduling scheme</td>
<td>64</td>
</tr>
<tr>
<td>4.3</td>
<td>Additional maximum TTL option field for deadline scheduling scheme</td>
<td>65</td>
</tr>
<tr>
<td>4.4</td>
<td>Topology used for experimentation</td>
<td>71</td>
</tr>
<tr>
<td>4.5</td>
<td>Congestion window growth rates in non-congested networks (Tested at 96kbps)</td>
<td>74</td>
</tr>
<tr>
<td>4.6</td>
<td>Congestion window growth rates in non-congested networks (Tested at 1024kbps)</td>
<td>74</td>
</tr>
<tr>
<td>4.7</td>
<td>Congestion window growth rates in non-congested networks (Tested at 5096kbps)</td>
<td>74</td>
</tr>
<tr>
<td>4.8</td>
<td>Congestion window growth rates in non-congested networks (Tested at 44736kbps)</td>
<td>74</td>
</tr>
<tr>
<td>4.9</td>
<td>Window growth rates in highly congested networks. Tested at 96kbps.</td>
<td>77</td>
</tr>
<tr>
<td>4.10</td>
<td>Window growth rates in highly congested networks. Tested at 1024kbps.</td>
<td>77</td>
</tr>
<tr>
<td>4.11</td>
<td>Window growth rates in highly congested networks. Tested at 5096kbps.</td>
<td>78</td>
</tr>
<tr>
<td>4.12</td>
<td>Window growth rates in highly congested networks. Tested at 44736kbps.</td>
<td>78</td>
</tr>
<tr>
<td>4.13</td>
<td>Throughput and RTT values from experiment comparing standard DCCP to deadline scheduling enabled DCCP. Tested at 96kbps.</td>
<td>81</td>
</tr>
<tr>
<td>4.14</td>
<td>Throughput and RTT values from experiment comparing standard DCCP to deadline scheduling enabled DCCP. Tested at 1024kbps.</td>
<td>82</td>
</tr>
<tr>
<td>4.15</td>
<td>Throughput and RTT values from experiment comparing standard DCCP to deadline scheduling enabled DCCP. Tested at 5096kbps.</td>
<td>83</td>
</tr>
<tr>
<td>4.16</td>
<td>Throughput and RTT values from experiment comparing standard DCCP to deadline scheduling enabled DCCP. Tested at 44736kbps.</td>
<td>84</td>
</tr>
<tr>
<td>4.17</td>
<td>Graph displaying reaction to 5 consecutive packet loss events in standard DCCP and deadline scheduling enabled DCCP. Tested at 44736kbps.</td>
<td>88</td>
</tr>
<tr>
<td>4.18</td>
<td>Graph displaying reaction to 5 consecutive packet loss events in standard DCCP and deadline scheduling enabled DCCP. Tested at 96kbps.</td>
<td>88</td>
</tr>
<tr>
<td>4.19</td>
<td>Graph displaying reaction to 5 consecutive packet loss events in standard DCCP and deadline scheduling enabled DCCP. Tested at 1024kbps.</td>
<td>88</td>
</tr>
<tr>
<td>4.20</td>
<td>Graph displaying reaction to 5 consecutive packet loss events in standard DCCP and deadline scheduling enabled DCCP. Tested at 5096kbps.</td>
<td>88</td>
</tr>
<tr>
<td>4.21</td>
<td>Graph displaying reaction to 5 consecutive packet loss events in standard DCCP and deadline scheduling enabled DCCP. Tested at 44736kbps.</td>
<td>89</td>
</tr>
<tr>
<td>4.22</td>
<td>Graph displaying reaction to 5% sustained packet loss in standard DCCP and deadline scheduling enabled DCCP. Tested at 96kbps.</td>
<td>91</td>
</tr>
<tr>
<td></td>
<td>Graph displaying reaction to 5% sustained packet loss in standard DCCP and deadline scheduling enabled DCCP. Tested at 1024kbps.</td>
<td>91</td>
</tr>
</tbody>
</table>

viii
4.23 Graph displaying reaction to 5% sustained packet loss in standard DCCP and deadline scheduling enabled DCCP. Tested at 5006kbps. 92
4.24 Graph displaying reaction to 5% sustained packet loss in standard DCCP and deadline scheduling enabled DCCP. Tested at 44736kbps. 92

5.1 RTT value for a normal DCCP flow showing increasing levels of congestion. 100
5.2 Simplified pseudocode for deletion purging mechanism. 102
5.3 Simplified pseudocode for pure ECN stale packet control mechanism. 106
5.4 Simplified pseudocode for phantom packet purging mechanism. 109
5.5 Simplified pseudocode for advanced deletion mechanism. 113
5.6 Simplified pseudocode for preventative hybrid discarding mechanism. 115
5.7 Single flow and multiple flow topology structures simulated for experimentation. 119
5.8 Simulated network topology components used in experimentation. 121
5.9 Variation in the number of "old" flows used in the simulated topologies. 122
5.10 Baseline fairness model used to measure fairness. No DCCP discard mechanism is enabled. 133
5.11 Baseline fairness model used to measure fairness. 134
5.12 Results used to measure fairness amongst streams initiated at the same time. Basic deletion mechanism active on the BOD. 136
5.13 Results used to measure fairness amongst streams initiated at the same time. Pure ECN mechanism active on the BOD. 137
5.14 Results used to measure fairness amongst streams initiated at the same time. Advanced delete mechanism active on the BOD. 137
5.15 Results used to measure fairness amongst streams initiated at the same time. Hybrid discard mechanism active on the BOD. 138
5.16 Fairness between streams starting at different times. No Discard mechanism active i.e. Normal DCCP operation. 140
5.17 Fairness between streams starting at different times. Basic delete mechanism active. 141
5.18 Fairness between streams starting at different times. Pure ECN mechanism active. 142
5.19 Fairness between streams starting at different times. Advanced deletion mechanism active. 143
5.20 Fairness between streams starting at different times. Hybrid discard mechanism active. 144
5.21 Send window rates for streams used to demonstrate fairness amongst dissimilar stream types. Basic delete mechanism active. 147
5.22 Send window rates for streams used to demonstrate fairness amongst dissimilar stream types. Pure ECN mechanism active. 148
5.23 Send window rates for streams used to demonstrate fairness amongst dissimilar stream types. Advanced delete mechanism active. 148
5.24 Send window rates for streams used to demonstrate fairness amongst dissimilar stream types. Preventative hybrid mechanism active. 149

6.1 Flow diagram showing categorization of packets logic. 160
6.2 Array structure used to store global variables needed to test TNTCN accuracy. 174
6.3 Logical queuing mechanism's flow directions. 180
6.4 First topology selected to test proposed mechanism. 188
6.5 Second topologies selected to test proposed mechanism. 189
6.6 Third topology selected to test proposed mechanism. 190

A.1 Chapter 6 experimental Topology 1. 206
A.2 Chapter 6 experimental Topology 2 . . . . . . . . . . . . . . . . . . . . . 207
A.3 Chapter 6 experimental Topology 3 . . . . . . . . . . . . . . . . . . . . . 208
A.4 Chapter 6 experimental Topology 4 . . . . . . . . . . . . . . . . . . . . . 209
A.5 Chapter 6 experimental Topology 5 . . . . . . . . . . . . . . . . . . . . . 210
A.6 Chapter 6 experimental Topology 6 . . . . . . . . . . . . . . . . . . . . . 211
A.7 Chapter 6 experimental Topology 7 . . . . . . . . . . . . . . . . . . . . . 212
A.8 Chapter 6 experimental Topology 8 . . . . . . . . . . . . . . . . . . . . . 212
A.9 Chapter 6 experimental Topology 9 . . . . . . . . . . . . . . . . . . . . . 213
A.10 Chapter 6 experimental Topology 10 . . . . . . . . . . . . . . . . . . . . 213
A.11 Chapter 6 experimental Topology 11 . . . . . . . . . . . . . . . . . . . . 214
A.12 Chapter 6 experimental Topology 12 . . . . . . . . . . . . . . . . . . . . 214
A.13 Chapter 6 experimental Topology 13 . . . . . . . . . . . . . . . . . . . . 215
A.14 Chapter 6 experimental Topologies 14, 15 and 16 (Link speeds varied
for each iteration. (Slow; Med; Fast)) . . . . . . . . . . . . . . . . . . . . . 215
A.15 Chapter 6 experimental Topologies 17, 18 and 19 (Link speeds varied
for each iteration.(Slow; Med; Fast)) . . . . . . . . . . . . . . . . . . . . . 216
A.16 Chapter 6 experimental Topology 20 . . . . . . . . . . . . . . . . . . . . 217
List of Tables

2.1 Calculations of transmit rate for slow start phase. .................. 31
3.1 RTT (in Milliseconds) experienced during experimentation with various fixed sizes queues. .................. 42
3.2 Packet loss rates experienced during experimentation. ............. 44
3.3 Goodput, throughput and loss ratio values recorded from the topologies using various fixed size queue settings. .................. 46
4.1 Window growth times in non-congested networks. .................. 76
4.2 Window growth times in highly-congested networks. ................ 79
4.3 Throughput, goodput and RTT rates. ................................. 85
4.4 Reaction to packet loss events during stabilized CA. ............... 89
4.5 Reaction to packet loss over a sustained period. .................... 93
5.1 List of topology groups and variations. ............................... 120
5.2 Throughput rates experienced during experimentation. ............. 126
5.3 Goodput rates experienced during experimentation. ................ 127
5.4 Packet ages experienced during experimentation. .................. 128
5.5 RTT values experienced during experimentation. .................. 130
6.1 Cisco EIGRP default delay metrics vs. actual delay metrics. .......... 163
6.2 Measured switching latency derived from simulation using a standard 1500 byte IPv4 frame. ................................. 166
6.3 Average percentage difference between calculated TNTCN and actual time recorded. ................................. 175
6.4 Percentage of packets assigned a TNTCN value higher than the actual time taken. ................................. 177
6.5 Queue depths recorded from simulation. ............................... 182
6.6 Average number of stale packets per workstation (per 1000 Packets). ................................. 184
6.7 Average age of packets. .............................................. 185
6.8 Ratio of stale to fresh packets from critical queue that arrived on time. ................................. 195
List of Abbreviations

Ack - Acknowledgment
AIMD - Additive increase multiplicative decrease
API - Application programming interface
AQM - Active queue management
BOD(s) - Bandwidth optimization device(s)
BT - Birth time
BW - Bandwidth
CA - Congestion avoidance
CBWFQ - Class based weighted fair queuing
CCID - Congestion control identifier
CCM - Congestion control mechanism
CNT - Current NTP time
CRC - Cyclic redundancy check
DCCP - Data congestion control protocol
DS3 - Digital signal 3 (44.736Mbps)
ECN - Explicit congestion notification
EDD - Earliest due deadline
EIGRP - Enhanced interior gateway routing protocol
ENET - Explicit NTP expiry time
EQ - Equation
FIFO - First in first out
GPS - Global positioning satellite
HC - Hop count
IETF - Internet engineering task force
IP - Internet protocol
<table>
<thead>
<tr>
<th>Acronym</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kbps</td>
<td>Kilobits per second</td>
</tr>
<tr>
<td>LD</td>
<td>Load</td>
</tr>
<tr>
<td>LLQ</td>
<td>Low latency queuing</td>
</tr>
<tr>
<td>M</td>
<td>Microseconds</td>
</tr>
<tr>
<td>MPEG</td>
<td>Moving picture experts group</td>
</tr>
<tr>
<td>MSL</td>
<td>Maximum segment lifetime</td>
</tr>
<tr>
<td>MTTL</td>
<td>Maximum time to live</td>
</tr>
<tr>
<td>MTU</td>
<td>Maximum transmission unit</td>
</tr>
<tr>
<td>NFT</td>
<td>No feedback timer</td>
</tr>
<tr>
<td>NTP</td>
<td>Network time protocol</td>
</tr>
<tr>
<td>P</td>
<td>Loss event rate</td>
</tr>
<tr>
<td>PBS</td>
<td>Probabilistic based scheduling</td>
</tr>
<tr>
<td>PDA</td>
<td>Personal digital assistant</td>
</tr>
<tr>
<td>PS</td>
<td>Packet size</td>
</tr>
<tr>
<td>Qos</td>
<td>Quality of service</td>
</tr>
<tr>
<td>QS</td>
<td>Queue size</td>
</tr>
<tr>
<td>RD</td>
<td>Reported delay</td>
</tr>
<tr>
<td>RED</td>
<td>Random early detect</td>
</tr>
<tr>
<td>RFC</td>
<td>Request for comment</td>
</tr>
<tr>
<td>RFS</td>
<td>Rate function scheduling</td>
</tr>
<tr>
<td>RTM</td>
<td>Real time media</td>
</tr>
<tr>
<td>RTO</td>
<td>Retransmission time out</td>
</tr>
<tr>
<td>RTP</td>
<td>Real time protocol</td>
</tr>
<tr>
<td>RTT</td>
<td>Round Trip time</td>
</tr>
<tr>
<td>S</td>
<td>Estimated packet size</td>
</tr>
<tr>
<td>SFB</td>
<td>Scholastic fair blue</td>
</tr>
<tr>
<td>SIP</td>
<td>Session initiation protocol</td>
</tr>
<tr>
<td>SS</td>
<td>Slow start</td>
</tr>
<tr>
<td>SSthresh</td>
<td>Slow start threshold</td>
</tr>
<tr>
<td>STDM</td>
<td>Statistical time division multiplexing</td>
</tr>
<tr>
<td>TCP</td>
<td>Transmission control protocol</td>
</tr>
<tr>
<td>TD</td>
<td>Total Delay</td>
</tr>
<tr>
<td>TDM</td>
<td>Time division multiplexing</td>
</tr>
</tbody>
</table>
TFRC - TCP friendly rate control
TLIQ - Time left in queue
TLTL - Time left to live
TNTCN - Time needed to cross network
TPS - Total packet size
TTL - Time to live
UDP - User datagram protocol
VoIP - Voice over Internet Protocol
WAN - Wide area network
WFQ - Weighted fair queue
X - Transmission rate
Xinst - Actual CCID3 transmit rate
Xprev - Previously calculated transmission rate
Xrecvremote - The receivers advertised transmit rate
Acknowledgments

I would like to thank my supervisors Mike Dixon and Terry Koziniec for being wonderful mentors, advisers and friends throughout this process. Without their encouragement, tireless advice and enthusiasm, this thesis would not have been possible, nor would the process of creating it have been nearly as enjoyable. I am truly grateful for all the time and effort they spent guiding and helping me during the past four years.

I would also like to give special thanks to Tanya McGill for her very kind assistance, support, encouragement and guidance during the final writing stages of the thesis. Having someone with Tanya’s expertise and knowledge was truly invaluable.

On a personal note I would like to thank Cedric Loeckx, Shahidah Ali, Brent Savage, Gareth Samuels and Nik Patel for their friendship and support. When the coding and writing became overwhelming, they were always there to provide friendly encouragement or a much needed break. Thank you all for being my family away from home and for being so understanding of the workload and associated stress.

Finally, I would like to thank my parents Ken and Colleen Wilson for their never ending support, belief in me, guidance and encouragement. I am truly blessed to have such wonderful parents and friends and am so grateful for all you do for me!
Chapter 1

Introduction

1.1 Background

Since the inception of the Internet, there has been a steady shift to port services such as television, radio and voice onto IP centric networks. While the advantages for doing this are numerous, these applications pose unique challenges to network engineers in that they are both bandwidth intensive and also extremely sensitive to network delays and bandwidth fluctuation.

As the push to move voice, video and other real time applications onto IP centric networks continues steadily, it is becoming evident that the current de facto Internet transport layer protocols, namely the Transmission Control Protocol (TCP) [1], and the User Datagram Protocol (UDP) [2], are not ideally suited to transporting real time traffic across the Internet. TCP is unsuitable due to the overheads it employs to ensure data reliability. One advantage of using TCP for real time data transfers however is its ability to adapt to changing conditions found on data networks through flow control and congestion avoidance mechanisms. These congestion mechanisms control the rate of transmissions and prevent applications overwhelming congested networks.

The alternative, UDP, encapsulates data packets using comparatively smaller headers but does not provide any form of reliability. As most real time applications do not require reliability or the large headers that are utilized by TCP to provide this reliability, UDP has emerged as the transport layer protocol of choice for transporting time sensitive data. In choosing UDP over TCP for the purposes of reducing overheads, applications forgo the congestion control mechanisms that would be afforded to them by the TCP protocol. As a result, it is possible for applications utilizing
A lack of congestion control or a suboptimal mechanism providing congestion control can lead to the over saturation of networks affecting not only the real time application but all other applications making use of network simultaneously. Given the fact that more and more real time services are likely to be ported onto data networks and that currently the de facto transport layer for these protocols is UDP, researchers are worried that improper congestion control techniques may lead to the eventual collapse of the Internet [3].

In order to overcome the limitations that both TCP and UDP face as well as the congestion control issues that arise when transporting real time data, the Data Congestion Control Protocol (DCCP) [4], was created to provide a more suitable transport layer protocol for real time data flows. Using a minimalistic approach similar to that of UDP, the DCCP protocol has small headers suited to real time small data packets such as those found in voice and video data frames. Another characteristic that DCCP and UDP share is that they are both unreliable and therefore do not retransmit lost packets.

In addition to this, DCCP also borrows certain features from TCP. The most notable feature being a variant of the mechanism TCP uses to perform data flow control and congestion avoidance. This prevents the DCCP flows from saturating network links in the event of congestion through the reduction of the send rate when loss events are detected. The combination of the best characteristics from both TCP and UDP, make DCCP an unreliable protocol that has low overheads and the ability to adapt to changing network conditions.

As DCCP only became an official standard in 2006 [4], it has yet to become as widely used as its predecessors TCP and UDP. However, with its intelligent design and suitability to real time online media, DCCP is perfectly poised to become a more widely used protocol as the demands for voice, video and other real time services increase on the Internet. Its authors Kohler and Floyd envisaged that one day DCCP may completely replace the UDP protocol [3].
1.2 Research problem

While DCCP is better suited to the transportation of real time data when compared to TCP and UDP, it fails to address the fact that real time data possesses a finite lifespan. Like UDP and TCP, DCCP treats all the packets it transfers across the network as equal irrespective of the type of data contained in those packets. To real time applications the data they receive is only useful for a finite period. If the data they receive elapses this useful period, in most cases it is simply dropped upon being received as it is no longer of use to the application as it is deemed stale. This research questions this design characteristic of DCCP and asserts that not all packets are necessarily equal. If the data within a packet is no longer useful to the destination application because it has become stale then that packet should not be treated in the same manner as a data packet carrying non stale data.

In DCCP networks, there is currently no mechanism for detecting when a packet becomes stale and therefore stale data is commonly transferred unnecessarily across the network. While the number of stale packets in a queue may only constitute a very small proportion of the total traffic in the queue, by simply being in the queue the stale packets add unnecessarily to the delay of every other packet in that queue that happens to fall behind them. As more and more real time packets begin to appear on networks, the number of stale packets on networks is likely to increase. It is simply not efficient to allow these stale packets to continue to traverse congested networks if it can be avoided.

1.3 Purpose of this research

The main purpose of this research is to remedy the research problem described above whereby stale packets are allowed by DCCP to traverse the network unabated. In order to achieve this, this research incorporates deadline scheduling into the DCCP standard. Deadline scheduling refers to the process by which data packets are ordered, prioritized or purged entirely from queues on intermediate network devices based upon the packet’s remaining lifespan value. The packet’s lifespan value is the amount of time, in milliseconds, the packet remains useful to the destination application before
it becomes stale. This value is dictated by the sending application when it passes the respective data down to the DCCP layer for transmission. This thesis describes in detail how the above mentioned deadline scheduling is incorporated into DCCP as well as the implications this has on the protocol in terms of additional overheads and performance degradation.

By incorporating the packet's life information directly into DCCP data carrying packets, the research shows how intermediate network devices known as Bandwidth Optimization Devices (BODs) are able to more efficiently transport real time data. In knowing how much time the data within a packet has remaining before it will become stale, a multitude of new packet prioritization techniques are made possible that are not currently supported by the DCCP standard. Specifically, by utilizing the newly added packet life information in the DCCP data packets, the research demonstrates how stale packets can be removed from the network by BODs in order to reduce congestion on the network. DCCP's unique design that provides no retransmission but still has congestion control based on drop events means the act of purging packets has very unique impact on a flow's characteristics when compared to either UDP or TCP.

Purging too many packets from a flow can cause unstable transmissions or premature flow timeouts if too many purge actions take place due to the lack of retransmission. By limiting the number of purge events per round trip period, it is demonstrated that such instability can be avoided.

Finally the research presents a novel probability based scheduling algorithm that uses CISCO's Enhanced Interior Gateway Routing Protocol (EIGRP) [5] routing table information to calculate a packet's probability of arriving at its intended destination within its useful lifespan. Based on this probability calculation packets are placed into one of three priority queues for transmission. The algorithm ensures that packets that need to be prioritized in order to prevent the packets from becoming stale are given priority in a manner that causes minimal detriment to the other packets traversing the network.

Through these mechanisms, the research contained in this thesis shows how the finite lifespan properties of real time data can be utilized to maximize scheduling efficiency and reduce the unnecessary transmission of stale packets that currently occurs in DCCP networks. With this additional deadline scheduling capability the suitabil-
ity of DCCP to real time applications is greatly increased. With this new capability, DCCP now has a distinct advantage over UDP. Armed with these additional capabilities, it is hoped the DCCP protocol will become the defacto transport layer of choice for real time application designers.

1.4 Research approach

In order to carry out the experimentation needed to explore the effectiveness and improvement that is possible through the addition of deadline scheduling in DCCP, a simulated network model was developed. This modeling and simulation was carried out using the Opnet Modeler simulation toolkit [6]. The ratified specifications/standards for implementing DCCP are those found in RFCs [3, 4, 7, 8, 9, 10].

Given the somewhat loose wording that occurs in RFC specifications with terms such as MAY or SHOULD, at times it was difficult to determine exactly how the these ambiguous terms should be implemented in code. The base code used to implement the standard DCCP CCID3 protocol in Opnet was written by Gu Xiao yuan in [11]. Modifications were made to this code in order to bring the code into alignment with RFC5762 [10]. RFC5762 specifies how RTP traffic is passed down to the DCCP layer from the application layer. These changes were made to ensure a correctly sized simulated RTP packet stream was created to utilize DCCP, and to ensure this was inline with the standard presented in [10]. The remainder of the baseline model created by [11] was left largely unchanged as the source code used appears to be ported from the Linux and NS2 versions of the DCCP code. This code has become the defacto code used when modeling DCCP in the research community and hence the decision to make use of it. A copy of the RFC5762 compliant Opnet Standard DCCP code used for this thesis is available at www.shadds.com/research/DCCP/.

Having created a functional and standard compliant implementation of DCCP in Opnet a modified variant was then created to incorporate deadline scheduling into DCCP. The first step in achieving this was to modify the DCCP endpoints to allow the application layer to pass the packet’s life information down to the transport layer. How this was best achieved as well as the impacts this had on the overall performance of the protocol were then explored. Once this deadline scheduling functionality was
added to the endpoints, intermediate devices (BODs) were then created to perform scheduling tasks based on the information stored in the DCCP option fields.

In the first implementation presented, the BOD's created were configured to only deal with DCCP data carrying packets that had existed beyond their useful lifespan as defined by the sending application. When the BOD detected a packet had become stale, it would then perform the necessary actions required to prevent unnecessary transfer of this redundant data across the network. While simply purging this packet upon detection was the obvious choice, this research shows that DCCP's unique design of having congestion control without retransmission provides a unique set of challenges to simply doing this. For this reason five varied approaches were explored and the success and/or lack thereof of each of these purging techniques was investigated.

In the second implementation, the BOD was configured to take a more preemptive approach to scheduling decisions. To do this the BOD uses EIGRP routing information to determine a packet's probability of arriving at its intended destination within its useful lifespan. Packets were then scheduled for transmission across the network based on this probability. This approach ensured more efficient delivery of packets and prevented endpoints from passing unrealistically low or high lifespan information to the DCCP layer. In addition if a packet was nearing the end of its useful lifespan and required prioritization in order to prevent the packet from becoming stale then BOD ensured this prioritization occurred at minimal expense to the other data packets traversing the network.

1.5 Organization of the thesis

This thesis is presented in seven chapters. In this chapter, a brief history of why DCCP was created is presented as background information. Following this, the research problem is introduced which explains how the current DCCP standard fails to take into consideration the finite nature of real time data. In addition, Chapter 1 also describes the research objectives and importance of the research carried out. It concludes with an overview of how the research presented in this thesis was carried out.

Chapter 2, presents a literature review covering two main areas of research. The
aim of this chapter is to show where the motivation for this research was derived from, as well as to provide an overview of research related to the work carried out in this thesis. This chapter will begin by exploring research in deadline based network scheduling which is followed by research focusing more specifically on packet life scheduling which is the particular area of deadline scheduling this research utilizes. Following this, the chapter summarizes the various DCCP standards and showcases the anatomy of a typical DCCP flow controlled by the Congestion Control Identifier 3 (CCID3) algorithm. This provides background into the inner workings of the protocol and provides an overview of DCCP for those not familiar with the various DCCP standards.

Chapter 3 introduces the simulation model that was designed and used to carry out the experimentation in this thesis. This chapter’s primary aim is to conduct experiments that investigate the effects placing fixed size queues on intermediate network devices have on DCCP performance. A secondary goal of this chapter is to ensure the standard DCCP model used for the experiments in chapters 4 - 6 operates in compliance with the various RFC’s that make up the protocol.

In Chapter 4, the DCCP model is adapted to include deadline scheduling. In order to achieve this additional option fields are appended to the standard DCCP Data packet header. These options are used to transport the data’s maximum age and birth time. The inclusion of these options creates the potential for a number of new scheduling mechanisms that are not currently possible in the current DCCP standard. As a result of these options, intermediate devices are able to determine how much time the data within a packet has remaining before it becomes stale. From this, the devices can then reorder their queues to ensure all packets are given the best chance of arriving at their destination within their useful lifespan. In addition to showing how the deadline scheduling information is added to the DCCP standard, this chapter also measures the detriment that is caused by the addition of the option fields through having larger headers in data carrying packets.

Chapter 5 presents five purging techniques that utilize the packet life information to remove packets that have exceeded their useful lifespan. DCCP has the unique characteristic of having congestion control but no retransmission of packets where loss events occur. For this reason the act of purging stale packets becomes non trivial as the effects of purging can be profound on a DCCP flow if not done correctly. This
Chapter investigates five packet purging techniques in order to highlight the effects purging can have on DCCP flow. Chapter 5 will also investigate the benefits and issues that emerge through the implementation of these mechanisms through proof of concept experimentation.

In Chapter 6 a novel cross layer Packet Life Scheduling mechanism is described that utilizes routing protocol information to determine a packet’s probability of arriving within its useful lifespan. This chapter will describe the design process of the mechanism and how it is implemented into the experiment model. Following this, experimentation to determine the benefits that can be deduced through this mechanism is carried out.

The final chapter (Chapter 7) draws on the findings from the various experiments and shows how the incorporation of deadline scheduling provides a solution to the research problem. This thesis demonstrates a new way in which the finite lifespan of real time data can be used to provide more efficient and timely transportation of real time media across future networks when DCCP is utilized.

1.6 Publications

TWO publications were generated using work carried out in this thesis to peer review and validate the research throughout the various stages. These publications are listed below.


Chapter 2

Literature Review

2.1 Introduction

This chapter is divided into two parts. In the first part research relating to deadline scheduling and then packet life scheduling is presented. This provides background to the mechanisms that are designed in later chapters (Chapters 4-6) that utilize the finite lifespan that real time data possesses to schedule packet transmission. Section 2.2 first describes work carried out in the broader area of deadline based network resource management. In particular this section focuses on research that addresses the finite lifespan properties that real time data possesses. Following this a more focused review of packet life discard schemes is presented in Section 2.3. Packet life discard schemes are a subset of deadline based resource management techniques and focus on removing packets that have exceeded their useful lifespan from queues or schemes where packets are prioritized based on their lifespan values. Section 2.3 explores research carried out using such schemes which form the basis of this research.

The second part of this chapter provides an overview of the areas in the DCCP standard that are applicable to this research. To achieve this, information is extracted from the various RFC’s specifications that are used to make up the protocol. These include RFC 4340 [4], RFC4342 [8], and RFC4341 [7], and RFC5762 [10]. Detailed semantics outlining how the protocol is designed and functions not directly used in this research have been deliberately omitted from this chapter in order to keep it concise. (Refer to [4] and [8] if more detailed information is required). The standards described are implemented directly into the simulated DCCP model used in experimentation carried out in this thesis. Following the review of the current DCCP standards, the anatomy of a typical DCCP flow utilizing CCID3 is presented to highlight the various
2.2 Deadline based network resource management

This section outlines work previously carried out in deadline based network resource management that is aimed at addressing the issues relating to the finite nature of real time data. The notion of data aging in real time applications is not a new concept and there have been numerous studies in the area such as those found in [12, 13, 14, 15, 16]. It can be argued that techniques such as Time Division Multiplexing [17] (TDM) and packet reservation schemes such as those found in [18], introduced the notion of packets possessing finite life periods. However, these techniques are somewhat out of the scope of what the packet switched lifetime mechanisms relevant to this research aim to achieve.

More specifically, deadline Based network resource techniques were the first to append and make use of time variables in data packets with the goal of prioritizing packets in a queue on intermediate devices such as routers to avoid those packets becoming stale. In a deadline based service, data packets are prioritized and transmitted according to a deadline-ordered scheduling policy. Utilizing information contained in the data frame such as the protocol type or timestamp, packets arriving at an intermediate device are sequenced for delivery based on certain attributes [19, 20]. In most cases these scheduling mechanisms maintain two or more simultaneous departure queues. Typically the primary queue is used to store time sensitive real time data packets and is serviced first when the device is deciding which data it should transmit. Once empty, after a certain period, or after a certain number of services, depending on the specification the other queues with lower precedence are serviced.

Wong and Liu present a novel approach to managing network resources in order to support real time applications in IP networks in [19]. In order to implement their proposed model they appended extra information to each real time packet at the network layer, signaling the absolute latest time a packet should be delivered by. Following this, they then implemented the simple deadline based channeling technique T/H, on a routing device placed between two endpoints [9]. The T/H scheduling
model essentially uses the packet’s time remaining to live (T) and the number of hops that remain until the destination (H) to determine the T/H value. To get the T/H value, the T value is simply divided by the H value. Packets with a smaller T/H value are considered more urgent than those with a larger T/H value and are therefore given precedence in the queue. Results from their simulation in a discrete simulator showed this scheduling methodology would ensure a higher percentage of on time packet deliveries when compared to a conventional first in first out (FIFO) queue on the routing device. Contrary to other techniques, Wong and Liu made use of only one queue for all traffic. Traffic that was not considered urgent was given an infinite maximum time to live and received very low precedence in the queue.

While the research shows their mechanism compares favorably to FIFO queue scheduling techniques, no comparisons to more advanced active queuing management (AQM) techniques such as Random Early Detect [21] (RED) or Stochastic Fair Blue [22] (SFB) were made. Based on their research it is not possible to determine how their model would perform compared to these more advanced queuing mechanisms. One other concern with Wong and Liu’s model is that there is no discussion relating to the security issues such a model could potentially encounter. With such a simplistic T/H scheduling technique, it would theoretically be possible for endpoints to append manipulated aging information in such a way that an application’s data packets could always be given priority in the queue. By continually specifying lower T/H values, it is possible for one miscreant endpoint to assume queue dominance.

In a similar study carried out by Figueira [20], the author proposes a solution where the queue deadline priority decision making process occurs on a routing device. Unlike in Wong and Liu’s work, Figueira’s proposed model does not allow the application layer to influence any decisions relating to queue prioritization as all decisions are made on the intermediate routing device using preconfigured classes. This alleviates the possibility of misbehaving endpoints which is significant; however, it means that there is no way for the application layer to influence the way in which intermediate devices determine priority of packets and therefore it is not a scalable solution. Regardless of the practical implementation issues surrounding this work, the research showed that by using a scheduling prioritization technique, the results obtained were favorable when compared to a simple FIFO scheduling technique.

In addition to the studies listed above, several other ways of implementing schedul-
ing policy for real time have been suggested and the most significant of these techniques include Delay-EDD [12] (Earliest Due Deadline), Jitter-EDD [14], Leave-in-Time [15], Rate-Function Scheduling [16] (RFS), VirtualClock [13], and $\Delta$-ordered scheduling [20]. Based on mathematical modeling and in some cases discrete simulation, these techniques all perform more efficiently than First-in-First-out (FIFO) scheduling.

Although deadline-ordered packet scheduling research first introduced the principles of finite life in real time data, the focus for the research was primarily situated in the field of Quality of Service (QoS) and Active Queue Management (AQM). The main emphasis of this work was to investigate how real time data could be given more definitive and guaranteed network conditions. Appending timestamp data into data packets and then being able to determine how much time a packet had left to live, was used to classify data and then prioritize its position in the queue accordingly. There are now a large number of implemented mechanisms that perform these functions, examples of which include Weighted Fair Queuing [23] (WFQ), Class Based Weighted Fair Queuing [23] (CBWFQ), Hierarchical Fair Service Curve [24], and ABE [25].

While improved QoS and AQM techniques became the natural progression for further research into deadline-ordered schemes, an alternative use for the maximum life information in a packet was introduced by Gurtov and Ludwig in [26]. Gurtov and Ludwig proposed that in addition to using a packet’s lifetime for queue prioritization, the information could be used to purge stale packets in transit on intermediate devices. The motivation for this work was driven by two reasons. Firstly transmitting stale data across a network wastes resources. Secondly, delivery of stale data leads to the delayed delivery of fresh data. In the following section, existing research focusing on using packet life in this manner is explored.

### 2.3 Packet life discard schemes

For the purposes of this thesis, the term packet life discard scheme is used to refer to any mechanism that uses a real time packet’s useful lifetime value to determine if the packet should be purged based on the time it has existed or is likely to exist for between endpoints.
As previously mentioned, Gurtov and Ludwig [26] are credited as being the first to introduce packet life discard schemes to the research community. In their model, each packet has a packet life variable embedded in a custom built IP header option field. Based on this information, packets that have existed beyond their useful time are purged on intermediate routers placed between the sender and the receiver. They also propose that packets not likely to be delivered within the valid time should also be purged, further increasing efficiency of bandwidth utilization. While they imply that it would be possible to use this packet lifetime discard mechanism in DCCP, all modeling carried out in their work makes use of TCP as well as a TFRC variant of TCP. One of the results of making use of TCP is that retransmissions of unacknowledged packets occur when stale packets are purged and no acknowledgment is received by the sender for the sent packet. Another problem with purging stale packets is the potential impact this has on congestion control mechanisms. When a stale packet is purged and no acknowledgment is received for the packet, the congestion control mechanism assumes the packet has been lost and reduces the congestion window.

To avoid the retransmission of purged packets as well as to avoid the false triggering of congestion events, Gurtov and Ludwig used a headercasting technique whereby packets that had become stale were stripped of the data portion of the packet, leaving only the header remaining. The header was then transmitted to the end device so the integrity of the transport layer remained intact. Upon receiving a complete header, the transport layer assumed the packet has been received and an acknowledgment was then sent to sender preventing a congestion event or retransmission event from occurring.

Testing of Gurtov and Ludwigs’ model showed that purging of stale packets on the network was able to improve overall performance for all applications utilizing the network. Specifically the reduction in the overall delay in delivery experienced by fresh packets when stale packets were purged was significant. Performance of the application for which packets were purged appeared to be unaffected by the purging process as the packet would likely have been dropped by the application once received.

Although Gurtovs and Ludwig’ model [26] was convincingly more efficient in resource utilization than standard TCP, the choice to append the packet life information at the network or IP layer introduces an issue whereby packets with extended IP options typically experience longer delays due to slow path processing on intermediate
routers than standard packets do. The additional efficiency that the scheme brings in
certain instances may be overshadowed by the added processing time it takes for each
packet to be processed on intermediate devices between the sender and the receiver.

While Gurtov and Ludwigs' study was and remains the most significant work in
the area, there have been a small number of other researchers who have attempted
similar research. One of these studies was carried out by Yuen and Yue [27] who
presented a scheme of purging and then subsequently prioritizing real time packets in
a data queue. The research was of a purely mathematical nature and no mention was
made regarding the practical implementation of the scheme. Their findings showed
that by purging stale packets in queues, other fresh packets in queue would experience
shorter delays and less contention for available bandwidth. Yuen and Yue [27] point
out that Gurtov and Ludwigs' model does not take into account varying transmission
delays occurring after the last hop intermediate device as would be expected if the link
to the endpoint was provided by an 802.11 wireless or third generation (3G) cellular
link. In Gurtov and Ludwigs' model a fixed delay value for the link between the last
hop router and the receiver is used to determine if a packet will arrive at the receiver
before it becomes stale. Using channel state information feedback about the link
from the Radio Network Controller, Yuen and Yue demonstrate how a more accurate
value for the delay on the last leg link can be derived. Determining this value allows
for more accuracy in the determination as to whether or not a packet will become
stale while in transit. While the mathematically calculated 10% increase in goodput
is questionable given the arbitrary delays that can occur in wireless networks, it is
indubitable that some benefit can be gained through the scheme.

Gurtov and Ludwigs' ideas were also used by Chebrolu's and Rao [28], who ex-
amined bandwidth optimization specifically for MPEG video where bandwidth was
severely restricted. By using a combination of Gurtov and Ludwig's packet discard
mechanism and knowledge of how MPEG video functions, they created Minimal Cost
Drop (MC Drop) [28]. MC Drop is used firstly to determine which packets already
are, or are likely to become, stale due to the high levels of congestion on the link.
Packets which are already stale are dropped immediately. Packets which are likely
to be become stale are then applied to a policy to determine which of these packets
should be dropped first to ensure the best video quality at the receiver side. The
results obtained from experiments showed that MC Drop allowed for higher overall
video quality via a severely restricted bandwidth link when compared to conventional non discard techniques.

In [29], Chen and Leung create a packet discard scheme with the aim of determining the probability of a real time packet arriving at the receiver device without becoming stale. To accomplish this they use the maximum allowable transfer delay a real time application can handle before the packet becomes stale. Based on this value, Gaussian distribution theory [30] is used to determine if the admitted packet can be delivered to the receiver within its useful lifetime with a certain probability. If it cannot be delivered within this time the packet is dropped from the queue. If the packet can be delivered within its useful time, the packet is left remaining in the queue and awaits transmission. Mathematical modeling of their scheme using ideal conditions showed significant improvement in reducing delay and queue sizes on the network.

TCP-RTM [31] was presented by Liang and Cheritan in 2002 and proposes a number of extensions that make TCP more suitable to real time data. One of these extensions involves marking stale packets to speed up the delivery of fresh packets. Radovanovic et al., in [32], took Liang and Cheritan’s [31] work one step further towards Gurtov and Ludwigs’ scheme [26] by using the SNOOP protocol to purge the packets identified by the TCP-RTM extension as being stale. In addition they implemented a mechanism for suppressing packet retransmission and congestion events resulting from the purging process. Results gathered from a NS2 simulation found that the packet discarding scheme improved overall network goodput and reduced the overall delay experienced by these packets when compared to conventional non discard techniques. This technique offers improved performance for real time applications using the TCP protocol and has much promise. The limitation with this scheme however is that very few real time applications make use of TCP given the high overheads it utilizes when compared to TCP.

2.4 Discussion of deadline scheduling in DCCP

All the above findings show unequivocally there is definite potential that can be gained by implementing a packet discarding scheme for real time traffic. There does not however appear to be a packet life discard scheme proposed or implemented
specifically for the DCCP protocol, a protocol which appears so well suited to such a scheme. Given that the DCCP protocol has been designed for the application of real time data, adding such a scheme could potentially be of major benefit to the protocol. Additionally, a number of problems which have hindered the widespread adoption of packet discarding schemes, such as the loss of congestion control integrity and having to suppress retransmissions, are removed in DCCP as it is an unreliable protocol. Using Gurtov and Ludwigs' packet discard principles, the research described in this thesis aims to design and implement an efficient and effective packet discard policy specifically for the DCCP protocol. By doing this, the protocol's utilization of available network resources is enhanced. Before describing how such a scheme would function however, the next part of this chapter will first introduce the DCCP protocol and some of the properties the protocol possesses that are relevant to this research.

2.5 Overview of the DCCP standards

The DCCP protocol is an unreliable protocol designed for long lived, delay sensitive data streams that require congestion control. DCCP is often described as either UDP with congestion control or TCP without reliability and the overheads that would be associated with this reliability. As DCCP is an unreliable protocol it does not retransmit packets that are lost or dropped. Even though DCCP is unreliable, it still makes use of an acknowledgment mechanism to detect packet loss in order to perform congestion control [3].

2.5.1 DCCP connection establishment, management and termination

In total DCCP uses 10 unique packet types for connection establishment, transfer of information between sender and receiver, connection termination, and sequence number resynchronization.

To initiate a connection, the DCCP protocol performs a connection establishment technique that closely resembles TCP's three-way handshake. Before data is sent
between endpoints, DCCP first exchanges three connection establishment packets. First, the sender transmits a DCCP Request to the receiver to initiate the connection. Upon receiving this, the receiver replies using a DCCP-Response message signaling it is ready for communication. When this DCCP-Response is received, the sender sends the DCCP-ACK message that will complete the three-way handshake and signal to the receiver that the next packet will contain data.

Once established, DCCP encapsulates application data into DCCP-Data packets. For the purposes of congestion control, the sender requires the receiver to acknowledge all DCCP-Data packets received. To send acknowledgments, the receiver utilizes one of two packets. If the receiver has no application data scheduled for delivery to the sender, it uses the DCCP-ACK to acknowledge the receipt of a DCCP-data packet. The DCCP-ACK packet is also called a pure ACK as it is used purely for the purposes of acknowledging data received and carries no application data. If the receiver has data scheduled for delivery to the sender, then it utilizes the DCCP-DataACK which is then piggy-backed on the data packet scheduled to go to the sender.

Once a sender has completed sending all the data required, a tear down process is initiated, again much like that found in TCP. The tear down process is comprised of either two or three steps. If the receiver wishes to terminate the connection, the process requires three steps. Firstly, the receiver signals its desire to terminate the connection by sending the sender a DCCP-CloseReq message. Upon receiving this, the sender then transmits a DCCP-Close message back to the sender. Once the receiver receives the DCCP-Close, it completes the termination process by sending a DCCP-Reset message back to the sender. If the sender wishes to terminate the connection, only two steps are required. First, the sender sends a DCCP-Close message to the receiver. The sender will respond with the DCCP-Reset signaling the connection has been terminated. In the event where abnormal termination of a connection occurs, it is possible for the DCCP-Reset message to be sent as a standalone step to terminate the connection.
2.5.2 Packet sequencing and numbering in DCCP

Every packet in DCCP contains a sequence number that is used to detect loss. As DCCP is unreliable, the detection of loss is used purely for congestion control and not for retransmission. DCCP’s sequence numbers vary from TCP in that unlike TCP sequence numbers, which are byte based and specify the next byte of data expected, DCCP sequence numbers are simply incremented by one each time a packet is sent. It is also important to note that the acknowledgment number in DCCP refers to the sequence number of the last received packet and not the sequence number of the next expected packet. This is further illustrated in the example in Figure 2.1.

![Diagram of DCCP sequence and acknowledgment numbering](image)

Figure 2.1: Example of DCCP sequence and acknowledgment numbering.

In DCCP the sequence number for a packet is incremented irrespective of whether the packet contains application data or not. As DCCP does not guarantee ordered delivery of data packets, it is possible for packets to arrive out of order and cause desynchronization of sequence numbers between endpoints. In the event that sequence number desynchronization occurs, DCCP-Sync and DCCP-SyncAck packet types are used to resynchronize the sequence numbers between endpoints.

2.5.3 DCCP packet format

In DCCP, the packet header is made up of four unique fields as shown in Figure 2.2 and is between 12 and 1020 bytes in length. Although there are 10 unique packets
utilized in DCCP, all DCCP packets must make use of the same initial generic packet header. Following this, and depending on the type of packet it is, each packet will then have a specific DCCP header portion used to perform the functions necessary for that packet type. Once these specific packet type values have been defined, the next component in a DCCP packet is the options portion, which is used to negotiate options and transfer option information between endpoints. The final portion of a DCCP packet is reserved for application data (if the data packet is used for data transfer). The specific elements found in each of the 10 unique packet types used in DCCP are detailed in [4].

![Figure 2.2: Components in a typical DCCP packet.](image)

2.5.4 Modularity in DCCP

One of the key design features of DCCP is its modular structure. When creating DCCP, the designers were very cognizant of the ever evolving and improving nature of congestion control techniques. In order to future-proof DCCP, the protocol was designed in such a way that future congestion control techniques can be added to the protocol seamlessly. To do this the DCCP protocol comprises of a single main module to which additional congestion control modules can then be added. The main module is responsible for tasks such as connection establishment, management and termination, as well as feature negotiation and cross layer communication. The pluggable congestion control mechanisms then specify how congestion control is applied
to the protocol. Exactly how congestion control is implemented in DCCP will be
discussed in the Section 2.5.6, however it is significant at this point to note that the
ability for different congestion control techniques to be used at any time gives DCCP
an advantage over TCP, which does not have the ability to negotiate or change its
congestion control mechanism as readily.

2.5.5 Feature negotiation options

DCCP endpoints use Change and Confirm options in the DCCP packet header to
negotiate and agree upon feature values used for communication between endpoints.
While feature negotiation always happens during the connection establishment phase,
feature negotiation can also take place at any time during the connection. One of the
features that is negotiated during connection establishment is the desired congestion
control mechanism. Other features that can be negotiated include ECN capabilities
and if and how a checksum is used during data transfer. In line with all the other
measures taken to ensure the protocol is future-proof, feature negotiation means that
new features can be added to the protocol at a later stage.

2.5.6 Congestion control in DCCP

Congestion control mechanisms can be divided into two main categories, AIMD (Add-
ditive Increase, Multiplicative Decrease) [33] or equation based [34]. As the name sug-
gests, AIMD congestion control mechanisms perform congestion control by increasing
the congestion window additively and then decreasing the size multiplicatively when
congestion is experienced. In the case of standard TCP implementations, the AMID
mechanism can decrease the congestion window by as much as half, which can lead to
arbitrary, jerky, bandwidth rates or the saw tooth phenomenon [35]. Equation based
congestion control mechanisms were developed to provide a smoother form of conges-
tion control to data streams using intricate mathematical equations to calculate the
amount to decrease or increase the congestion window based on network conditions
and congestion events [36].

While past inferences have been made as to which category of control mecha-
nism is superior, the general research community now commonly accepts that the different techniques are not directly comparable. AMID approaches to congestion control are able to rapidly utilize available bandwidth but are at times over zealous in their congestion backoff approach and therefore can introduce arbitrary delay to data communications and “jerky” throughput. Therefore AMID based congestion control mechanisms are ideal for short, bursty traffic that is not negatively affected by arbitrary delays [4].

Equation based mechanisms are typically slower in making use of available bandwidth and reacting to network condition changes, but at the same time are less radical in their approach to reducing the congestion window once it has grown [4]. This means that there is a smoother more deterministic throughput of data between endpoints. Equation based congestion control is more suited to long lived data transfers where the application uses a fixed size frame and requires a specific rate of data flow. These requirements are typical of those required by Voice over IP protocols such as G.711[37].

In DCCP, congestion control is provided by plug-in modules called Congestion Control Modules (CCM). The CCM specifies how the sender should respond to congestion events and how the sender, intermediate devices and receivers should signal such events. Each CCM employed by the protocol is assigned a unique number called the Congestion Control Identifier (CCID) which is issued by the IETF DCCP workgroup. When a connection is established between two endpoints, negotiation of the desired CCID takes place for each half of the connection.

It is possible for a single end-to-end connection to use a different CCID for each traffic flow direction. For example traffic flowing from A => B could make use of CCID2, while traffic flowing from B => A makes use of CCID3. The ability for DCCP to negotiate the desired CCID for the connection allows an application to choose a congestion control mechanism best suited to its requirements. Currently there are three CCIDs available for applications to utilize when using DCCP. These include CCID2 - TCP Like DCCP [7], CCID3 - TFRDCCP [8] and CCID4 - TFR variant with support for small packets DCCP [9].

CCID2 which is also known as TCP-Like DCCP[7] is classified as an AMID based congestion control technique. Using additive increases and multiplicative decreases, the protocol aims to transport as much data as possible in the shortest possible time. In addition, the protocol is also able to react to changes in network conditions rapidly.
As it uses the same congestion algorithm as TCP, CCID2 congestion control is prone to the same phenomena that are present in TCP. These include the introduction of arbitrary delays and jitter during connections. CCID2 is recommended for use where the application requires a short lived burst of traffic or where the application uses varying packet sizes [7].

The congestion window growth in CCID2 is essentially the same as that found in typical modern TCP variants utilizing selective acknowledgments (SACK). Once a connection is established CCID2 enters a phase known as slow start, allowing the protocol to rapidly make use of any available network bandwidth. In traditional TCP the congestion window is typically increased by one packet for each round trip time (RTT), however CCID2 uses a more modern TCP variant outlined in [35]. By using this modified variant CCID2 is able to provide a much more aggressive method of utilizing available bandwidth when compared to the previous techniques [35]. Using a variable known as the acknowledgment ratio (Ack_Ratio), DCCP exponentially increases the size of the congestion window. Once the congestion window reaches a threshold known as the slow start threshold (ssthresh), the protocol then enters a mode called congestion avoidance.

During the congestion avoidance mode, the congestion window is increased by one packet for every window of data acknowledged without lost or marked packets. This results in linear growth of the congestion window rather than the exponential growth found in the slow start phase. During the congestion avoidance phase, a congestion event will result in the congestion window being halved.

CCID3 or TCP Friendly Rate Control (TFRC) [8], uses an equation based congestion algorithm to provide smooth throughput by avoiding the abrupt changes that are a characteristic of TCP and TCP-Like DCCP. CCID3 was derived from the TCP friendly congestion control mechanism (TFRC) described in [36] and is ideal for streaming media applications that use minimal buffering before playback. While the overall throughput is smoother in CCID3, the algorithm will not utilize all the available bandwidth as quickly as TCP and is not suited to applications that vary packet size. In addition CCID3 is also slower to react to changes in network conditions.

Like CCID2, CCID3 begins building the congestion window in the slow start phase. However in CCID3 the congestion window size is doubled every time an acknowledgment is received. During this slow start phase exponential growth will
continue until a packet drop occurs or an acknowledgment is returned that has been marked by the ECN protocol. When this occurs, the sender will stop doubling the congestion window for each acknowledgment message and will use the loss event rate variable to determine the congestion window size.

The loss event rate variable is calculated by the receiver using packet sequence numbers and is fed back to sender via the loss interval header option in acknowledgment packets. The loss event rate is derived firstly by calculating the number of lost packets as a fraction of the total number of packets transmitted. This value is then weighted over several previously calculated loss event rates (up to the last 28 last calculated loss event rates) to determine the new loss event rate. Upon receiving a message with a loss event rate value, the sender then calculates the current RTT using the acknowledgment packet. The loss event rate and RTT value are then fed into TFRC’s throughput equation, which will produce the acceptable transmit rate. Once the acceptable transmit rate has been determined, the sender then adjusts its transmit rate to this newly calculated acceptable transmit rate. As the loss event rate is weighted over a number of calculations, CCID3 provides a less responsive but smoother congestion control technique for traffic flows.

The final congestion control technique available to DCCP at the time of writing this thesis is CCID4 (defined in [9]). CCID4 is an equation based congestion based congestion control mechanism and is designed for applications that make use of small packet sizes or for applications that vary their send rate by adjusting the packet size at the application layer. CCID4 makes use of an experimental small-packet-friendly variant of the TFRC mechanism used by CCID3 [9]. At present CCID4 is still in the experimental stages.

As CCID3 is the most suitable for the real time applications targeted by this research it is used as the congestion control mechanism for the remainder of the thesis. In the next section (Section 2.6) a more detailed overview of CCID3 is given. In addition the anatomy of a typical DCCP flow controlled by CCID3 is presented to highlight the factors that influence the DCCP send rate.
2.6 Anatomy of a typical CCID3 controlled session

This section describes and demonstrates the behavior of standard DCCP when CCID3 and RTP are used in conjunction with the protocol. The various phases that occur during a typical DCCP CCID3 flow are presented as well as the events that trigger the transition between these phases. During this explanation, four key factors that affect DCCP performance will be introduced and discussed. These four factors govern the transmission rate that occurs during a DCCP data flow.

2.6.1 The function of CCID3 in DCCP

The ultimate purpose of any congestion control algorithm is to control the rate at which data is transmitted into the network to ensure maximum utilization of bandwidth whilst also ensuring minimal packet loss. To achieve these goals CCID3 makes use of a variable known as an allowed transmit rate (known herein as transmit rate). The transmit rate specifies how much data can be sent to the receiver, before an acknowledgment for the data indicating it has been received is expected from the receiver. RFC4342 [8], specifies that the CCID3 receiver is to send an acknowledgment packet back to the sender at least once per round-trip time, unless the sender is sending at a rate of less than one packet per round-trip time. Failure to receive an acknowledgment for a segment of data is interpreted by CCID3 to mean that packet loss has occurred on the network and that the send rate should be reduced to prevent further packet loss.

One very important thing to note is that the CCID3 transmit rate is not necessarily the rate at which actual transmission of packets will occur at. If the CCID3 transmission rate exceeds the rate that the physical transmission interface on the sender is capable of sending at then queuing will occur and the rate will be governed by the transmission speed of the interface rather than the CCID3 transmit rate. If the interface is capable of transmitting the packets at a rate faster than CCID3 specifies, then the CCID3 transmission rate will govern the speed of the transmission.
2.6.2 Phases of a CCID3 flow

Figure 2.3 shows an example of the transmission rate experienced in a typical DCCP flow between two hosts communicating using the RTP and DCCP protocol. As the aim of this section is to show the distinct different phases that occur during a CCID3 session, the topology used to procure this example utilizes 96kbps links that are placed between the sender and the receiver. These slow speed links were selected to force each phase to have a longer duration making it easier to identify as well as making it easier to distinguish the events that trigger the various phases. The example figure highlights the two phases in which CCID3 operates. In the first phase (indicated by red), the CCID3 algorithm enters a mode known as slow start mode. As can be seen there is a rapid growth in the transmission rate during this stage. In the second phase (indicated by yellow) the CCID3 algorithm enters an equation based throughput mode. During this mode, there is a smoother less oscillating transmission rate. Please note that this flow is based on a RTP session utilizing an elastic G.711 stream to highlight the nuances of the DCCP protocol. Different protocols/ codecs and applications that operate in an elastic mechanism may result in different flow patterns. These are however beyond the scope of this thesis.

![Figure 2.3: Typical DCCP behavior. Transmission rate.](image-url)
2.6.3 Slow start phase

When a CCID3 session is first initiated, DCCP has no knowledge of the network speeds and/or network conditions that exist between the communicating devices. Like TCP, the DCCP protocol starts in a mode known as slow start. This mode probes network conditions by allowing the transmit rate to grow rapidly in order to utilize the maximum bandwidth available. In order to achieve this rapid growth in the transmit rate, the sender will roughly double its allowed sending rate each RTT[8]. The transmission rate will continue to increase exponentially until a packet drop event occurs or, if ECN is enabled, when a packet is received with an ECN marking indicating the transmit rate should be reduced. Upon detection of a loss event or an ECN marked packet, the sender ends the slow start phases and uses the loss event rate (equation based throughput mode) to calculate the transmit rate [36].

During slow start, the initial CCID 3 transmit rate is set to at least two packets and at most four packets [8]. The number of packets sent in the initial RTT is bound by packet size passed down from the application layer. In the event that three or four packets are sent in the initial RTT, the DCCP segment must not exceed 4380 bytes per RTT. After this initial value, the number of packets sent per RTT will roughly double every RTT until the slow start phase has ended (packet loss occurs). Simply stating the transmit rate will double every RTT during slow start is slightly ambiguous and as will be shown not necessarily always true. Figure 2.4 shows a magnified version of the slow start phase.
In Figure 2.4, it can be seen that the data transfer session is initiated at 5 seconds. As no packet loss has occurred, the connection commences in slow start mode and remains in this mode until 11.933 seconds when the first packet loss is reported by the receiver. At this point, slow start mode ceases, the transmission rate is halved due to the reported loss, and the equation based weighted throughput algorithm for TFRC [36] assumes control of calculating the transmission rate. During slow start, the transmission rate is calculated using the formula provided in Equation 2.1.

\[
X = \max(\min(2 \times X_{\text{prev}}, 2 \times X_{\text{recvremote}}), \left(\frac{S}{\text{RTT}/10000.0}\right)); \quad (2.1)
\]

Where:

- \(X\) = Transmission Rate.
- \(X_{\text{prev}}\) = Previously calculated transmission rate.
- \(X_{\text{recvremote}}\) = The receivers advertised transmit rate.
- \(S\) = The estimated calculated packet size.
- \(\text{RTT}\) = The estimated calculated RTT value.

As shown in the slow start formula (Eq.2.1), the transmission rate can be obtained...
from one of three sources. It can be either the previous transmit rate (Xprev), the last reported transmit from the receiver (Xrecvremote) or the packet size(s) divided by the reported RTT. In order to calculate the transmit rate, first the previous transmission rate and the last reported remote receive rate are doubled and the smaller of these values is selected. This value is then applied to a MAX function to determine if it is larger than the value calculated by dividing the estimated packet size(s) by the estimated RTT. Whichever value is larger is used as a candidate transmission rate. Most literature states this is how the transmission rate is calculated for CCID3 during slow start however this is erroneous as evident by the sudden drop that is seen in the transmission rate at 6.103 seconds in Figure 2.4. Although not compulsory, the RFC4342 [8] standard strongly recommends the transmission rate be used in conjunction with an anti-oscillation formula to prevent over saturation from occurring on a network during the slow start phase.

2.6.3.1 Preventing oscillations

The anti-oscillation formula described in RFC3448 [36] and RFC4342 [8] allows the sender to detect increases in delay which in most cases signify an increase in queue size and increase in congestion. Both of these occur when the transmit rate nears the maximum possible value for the link it is probing. When the RTT value begins to increase excessively, indicating the impending maximum transmission rate is nearing, the anti-oscillation mechanism will begin to lower the transmit rate to prevent excessive data being placed into an overly congested network.

If this mechanism was to be omitted, the congestion window would continue to increase exponentially until it exceeded the link capacity forcing a loss or timeout event to occur. This leads to scenarios where a large number of packets are sent onto an oversaturated link and are therefore likely to be lost or at very least subjected to excessive delay. The anti-oscillation mechanism still allows the necessary loss event to occur but reduces the number of packets that could be lost due to over saturation of the network by reducing the transmission rate.

To prevent oscillations from occurring during the slow start phase, the sender maintains an average of RTT values (Rsqmean) throughout the duration of the slow start mode. The Rsqmean provides an exponentially weighted moving average of
RTT values to which the last calculated RTT can be compared to, to determine how much the RTT has increased [36]. To calculate the Rsqmean value the sender uses the formula found in Equation 2.2.

\[
Rsqmean = (0.9 \times \text{Prevrsqmean}) + (0.1 \times \sqrt{\text{Rttcurrent}}); \quad (2.2)
\]

Where:
- \text{Prevrsqmean} = The previous R-sqmean value.
- \text{RttCurrent} = The current RTT time.

Once the Rsqmean is calculated, it is factored into the transmit rate in order to prevent oversaturation of the network. When the anti oscillation mechanism is enabled, the ultimate transmission rate calculation for a CCID3 session during slow start uses the formula shown in Equation 2.3.

\[
X_{\text{inst}} = X \times \left( \frac{Rsqmean}{\sqrt{\text{Rttcurrent}}} \right); \quad (2.3)
\]

Where:
- \(X_{\text{inst}}\) = The actual transmit rate used by CCID3 for communication.
- \(X\) = Candidate Transmit rate from: \(\max\) (\(\min\) (2*X\_prev, 2*X\_recv\_remote), (s/RTT)).
- \(Rsqmean\) = The RTT average value.
- \(Rttcurrent\) = The current estimated of the RTT value.

## 2.7 Demonstration of transmit rate calculations during slow start

To show the effects the above equations have on the slow start transmission rates, an example of a single CCID3 controlled flow will now be presented. The results below were derived from the simulation model used in all experimentation throughout this thesis. The reason for demonstrating these results is twofold. Firstly to highlight the nuances of CCID3 and secondly to validate that the model is functioning in accordance with the various standards. Figure 2.5 shows the transmission rate, the RTT and queue size respectively.
As can be seen in the graphs 2.5 a typical CCID3 controlled transmit rate experiences rapid and exponential growth when a session is initiated. This occurs as there is no contention for network resources as the outgoing queue on the transmit interface is empty and therefore RTT times are optimal. As the transmit rate continues to increase, the queue on the outgoing interface begins to grow. The time that the packets wait in the queue adds to the total RTT. As can be seen in Figure 2.5, there is clear correlation between the increase in queue size and the increase in the reported RTT. As the RTT size begins to increase, the anti oscillation mechanism reacts by reducing the transmit rate. The effects of the anti oscillation mechanism are most visible at approximately 6 seconds when there is an actual decrease in the transmit
rate. It is important however to note that the mechanism is functional throughout the slow start mode. The sudden drop in transmit rate seen at 11.274 seconds is due to packet loss. This occurs because the queue exceeds the maximum size threshold value and results in a packet loss event. The delay between the time when the queue reaches maximum capacity and when the sender reacts to packet loss occurs as the receiver is responsible for reporting the loss event. This variation is due to the time it takes for the sender to received an acknowledgment reporting the packet loss event.

In Table 2.1 the complete results that were used for this example are shown. These demonstrate how CCID3 calculates the final transmission rate ($X_{\text{inst}}$) used during the slow start phase.

<table>
<thead>
<tr>
<th>Time (sec)</th>
<th>RTT</th>
<th>X</th>
<th>$X_{\text{recv}}$</th>
<th>$S/RTT$</th>
<th>MAX($\min(2x,2)$, $X_{\text{recv}}/S/RTT$)</th>
<th>R$_{\text{qmean}}$</th>
<th>R$_{\text{sample}}$</th>
<th>$X_{\text{inst}}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.0000</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>NA</td>
<td>1945.032</td>
<td>0.21633</td>
<td>0.0468</td>
<td>1.840</td>
</tr>
<tr>
<td>5.2500</td>
<td>0.0946</td>
<td>1840</td>
<td>0.1945.032</td>
<td>1945.032</td>
<td>1945.031712</td>
<td>0.21633</td>
<td>0.0468</td>
<td>1.840</td>
</tr>
<tr>
<td>5.3500</td>
<td>0.0987</td>
<td>2051</td>
<td>1951.282</td>
<td>1951.282</td>
<td>1951.282051</td>
<td>0.21633</td>
<td>0.0468</td>
<td>2.051</td>
</tr>
<tr>
<td>5.4397</td>
<td>0.0853</td>
<td>3680</td>
<td>2157.096</td>
<td>3680</td>
<td>2157.096</td>
<td>0.21633</td>
<td>0.0468</td>
<td>3680</td>
</tr>
<tr>
<td>5.4957</td>
<td>0.0813</td>
<td>4102</td>
<td>2263.223</td>
<td>4102</td>
<td>2263.223</td>
<td>0.216356</td>
<td>0.0469</td>
<td>4098.060</td>
</tr>
<tr>
<td>5.6243</td>
<td>0.0777</td>
<td>763</td>
<td>2368.082</td>
<td>754</td>
<td>2368.082</td>
<td>0.214377</td>
<td>0.0469</td>
<td>7747.3005</td>
</tr>
<tr>
<td>5.6718</td>
<td>0.0745</td>
<td>8196</td>
<td>2469.799</td>
<td>8196</td>
<td>2469.799</td>
<td>0.216396</td>
<td>0.0469</td>
<td>8199.6267</td>
</tr>
<tr>
<td>5.7642</td>
<td>0.0716</td>
<td>9642</td>
<td>2569.832</td>
<td>9592</td>
<td>2569.832</td>
<td>0.216389</td>
<td>0.0468</td>
<td>15928.1472</td>
</tr>
<tr>
<td>5.8706</td>
<td>0.0701</td>
<td>9896</td>
<td>2621.083</td>
<td>9738</td>
<td>2621.083</td>
<td>0.21902</td>
<td>0.0599</td>
<td>16188.1441</td>
</tr>
<tr>
<td>6.0130</td>
<td>0.0738</td>
<td>2028</td>
<td>2492.225</td>
<td>29028</td>
<td>2492.225</td>
<td>0.22093</td>
<td>0.1076</td>
<td>44668.9432</td>
</tr>
<tr>
<td>6.1510</td>
<td>0.0821</td>
<td>20928</td>
<td>2241.169</td>
<td>29028</td>
<td>2241.169</td>
<td>0.246577</td>
<td>0.1572</td>
<td>13015.7265</td>
</tr>
<tr>
<td>6.2623</td>
<td>0.0937</td>
<td>20928</td>
<td>1963.714</td>
<td>29028</td>
<td>1963.714</td>
<td>0.264629</td>
<td>0.1999</td>
<td>12480.4151</td>
</tr>
<tr>
<td>6.4398</td>
<td>0.1078</td>
<td>20928</td>
<td>1706.865</td>
<td>29028</td>
<td>1706.865</td>
<td>0.288495</td>
<td>0.2355</td>
<td>12441.4210</td>
</tr>
<tr>
<td>6.5720</td>
<td>0.1232</td>
<td>20928</td>
<td>1493.506</td>
<td>29028</td>
<td>1493.506</td>
<td>0.31087</td>
<td>0.2624</td>
<td>12700.8174</td>
</tr>
<tr>
<td>6.7178</td>
<td>0.1393</td>
<td>20928</td>
<td>1320.899</td>
<td>29028</td>
<td>1320.899</td>
<td>0.333387</td>
<td>0.2852</td>
<td>13056.9381</td>
</tr>
<tr>
<td>6.8557</td>
<td>0.156</td>
<td>20928</td>
<td>1179.487</td>
<td>29028</td>
<td>1179.487</td>
<td>0.35533</td>
<td>0.3076</td>
<td>13468.0838</td>
</tr>
<tr>
<td>7.0645</td>
<td>0.1748</td>
<td>20928</td>
<td>1052.632</td>
<td>29028</td>
<td>1052.632</td>
<td>0.378457</td>
<td>0.3441</td>
<td>13502.1277</td>
</tr>
<tr>
<td>7.2626</td>
<td>0.1959</td>
<td>20928</td>
<td>939.2347</td>
<td>29028</td>
<td>939.2347</td>
<td>0.402805</td>
<td>0.3868</td>
<td>13536.3262</td>
</tr>
<tr>
<td>7.4992</td>
<td>0.2196</td>
<td>20928</td>
<td>837.8871</td>
<td>29028</td>
<td>837.8871</td>
<td>0.428335</td>
<td>0.4324</td>
<td>13621.2453</td>
</tr>
<tr>
<td>7.7696</td>
<td>0.2471</td>
<td>20928</td>
<td>744.3364</td>
<td>29028</td>
<td>744.3364</td>
<td>0.45595</td>
<td>0.4963</td>
<td>13546.7984</td>
</tr>
<tr>
<td>8.0687</td>
<td>0.2788</td>
<td>20928</td>
<td>659.9713</td>
<td>29028</td>
<td>659.9713</td>
<td>0.485481</td>
<td>0.5644</td>
<td>13524.0491</td>
</tr>
<tr>
<td>8.4275</td>
<td>0.3154</td>
<td>20928</td>
<td>583.3682</td>
<td>29028</td>
<td>583.3682</td>
<td>0.517301</td>
<td>0.6459</td>
<td>13470.6475</td>
</tr>
<tr>
<td>8.7776</td>
<td>0.3563</td>
<td>20928</td>
<td>516.4187</td>
<td>29028</td>
<td>516.4187</td>
<td>0.550765</td>
<td>0.7258</td>
<td>13529.5162</td>
</tr>
<tr>
<td>9.1902</td>
<td>0.4027</td>
<td>20928</td>
<td>456.9158</td>
<td>29028</td>
<td>456.9158</td>
<td>0.58632</td>
<td>0.8214</td>
<td>13538.9417</td>
</tr>
<tr>
<td>9.6205</td>
<td>0.4539</td>
<td>20928</td>
<td>405.3756</td>
<td>29028</td>
<td>405.3756</td>
<td>0.623385</td>
<td>0.9158</td>
<td>13632.7588</td>
</tr>
<tr>
<td>10.1064</td>
<td>0.5112</td>
<td>20928</td>
<td>359.9374</td>
<td>29028</td>
<td>359.9374</td>
<td>0.662388</td>
<td>1.027</td>
<td>13679.0300</td>
</tr>
<tr>
<td>10.65794</td>
<td>0.575</td>
<td>20928</td>
<td>320</td>
<td>29028</td>
<td>320</td>
<td>0.703419</td>
<td>1.1507</td>
<td>13723.3783</td>
</tr>
<tr>
<td>11.2747</td>
<td>NA</td>
<td>20928</td>
<td>284.5214</td>
<td>29028</td>
<td>284.5214</td>
<td>0.746783</td>
<td>1.2929</td>
<td>13744.8449</td>
</tr>
</tbody>
</table>

Table 2.1: Calculations of transmit rate for slow start phase.

2.7.1 Equation based flow control phase

Once a packet loss event is detected, the slow start mode is immediately ended and the transmission rate halved. From this point onwards, the equation based flow control phase is entered and the TRFC throughput equation is used to calculate
the transmission rate. In Figure 2.6 this transition occurs at 11.933 seconds. Once DCCP enters the equation based flow it will remain in this phase until the end of the transmission.

Figure 2.6: Transition from slow start phase to equation based flow control.

One of the key variables used by the TFRC equation to calculate the transmit rate is the loss event rate variable. This variable is derived from the loss event rate reported in acknowledgments received back from the receiver and specifies the number of loss events that have occurred as a fraction of the total number of packets transmitted during that same period. This value is then weighted over several previously calculated loss event rate values. The loss event rate, a round-trip time estimate, and the average packet size are then placed into the CCID3 throughput equation to
calculate the transmit rate [8]. Like much of CCID3, this equation is borrowed from RFC3448 [36]. Equation 2.4 shows this formula.

\[
X = \frac{s}{(R \sqrt{(2 \times b \times p/3)} + (t - RTO \times (3 \times \sqrt{(3 \times b \times p/8)}) \times p \times (1 + 32 \times p^2)))}
\]  

(2.4)

Where:
- \(X\) = is the transmit rate in bytes/second.
- \(s\) = is the packet size in bytes.
- \(R\) = is the round trip time in seconds.
- \(p\) = is the loss event rate, between 0 and 1.0, of the number of loss events as a fraction of the number of packets transmitted.
- \(t\) = RTO: is the TCP retransmission timeout value in seconds.
- \(b\) = is the number of packets acknowledged by a single TCP acknowledgment.

Derived from RFC3448 [8].

2.7.1.1 Calculating the loss rate

During a CCID3 controlled session, the receiver maintains a list of packets that have arrived to determine if any packets are missing. In order to do this, the receiver maintains a list of packet sequence numbers for the data packets that have arrived and their respective timestamps. From this the receiver is able to determine which packets are missing. Like TCP, CCID3 receivers consider a packet loss to have occurred when three packets with a higher sequence number are received. Unlike TCP however, CCID3 will accept late packet arrivals (3 or more packets late) and not report these as missing if the packet arrives within the same RTT period. Loss events are then reported back to the sender in acknowledgment packets.

2.7.2 CCID3 Reaction to Packet loss

Throughout a CCID3 rate controlled session, the throughput rate will be constantly recalculated by the sender when new acknowledgment packets arrive. When packet loss occurs, these losses will be reported back to the sender via acknowledgment from the receiver. Once the acknowledgment is received, the loss events are placed into the CCID3/TFRC algorithm and a new transmit rate is derived. As the loss event rates are weighted over several previous loss event rates, the transmit rate will remain less reactant to abrupt network changes when compared to CCID2 and TCP when packet
loss occurs. This is desirable for applications that wish to have reduced jitter and more constant transfer rates.

2.7.2.1 CCID3 Reaction to non acknowledgment of data

As seen in Equation 2.4, the packet loss information found in an acknowledgment plays a vital role in the determination of the transmit rate. As DCCP does not use a reliable mechanism to ensure acknowledgments are in fact received, the protocol uses a safeguard to reduce the transmit rate in the event that an acknowledgment packet is lost or never transmitted to the sender from the receiver. To achieve this DCCP makes use of a timer known as the nofeedback timer [8].

RFC4342 [8] states that if an acknowledgment is not received from the sender within the time specified by the nofeedback timer, the sender is to halve its transmit rate. The sender sets the nofeedback timer to at least four round-trip times, or to twice the interval between data packets, whichever is larger. This formula is shown in Equation 2.5.

$$NFT = \max((4 \times RTT), (2 \times \frac{PktSize}{TX - Rate}))$$

(2.5)

Where:

- NFT: is the No Feedback timer value.
- RTT: is the current Round Trip Time value.
- PktSize: is the Size of the packet in bits.
- TX-Rate: is the current transmit rate in bits per second.

In the event that acknowledgments are not received and the send rate is halved continually for an extended period, the allowed sending rate is never to be reduced to a rate below one packet per 64 seconds [8].

2.7.2.2 Reaction to DCCP slow receiver

In anticipation of the types of low powered end user devices that may make use of the protocol, such as PDA devices and cellular telephones, the designers of DCCP
included an option that allows the receiver to signal to the sender that it is having difficulty servicing the sender’s packets [8]. The use of the Slow Sender option explicitly requests that the sender is not to increase its send rate even though no packets have been dropped. If the receiver wishes for this rate to be maintained over a long period of time, the Slow Receiver option must be sent in every acknowledgment for the duration which the receiver wishes the rate to remain the same.

If the receiver wishes for the rate to be reduced it must either use the Drop Code 2 ("receive buffer drop") or create a fictitious ECN Nounce indicating the sender should slow down. If the sender places a Drop Code 2 in an acknowledgment, the sender is to reduce the transmit rate by one packet per RTT for each acknowledgment received with the Drop Code 2 option. The transmit rate can only be reduced to a minimum of 1 packet per RTT. Forging an ECN Nounce to indicate congestion has occurred could also be used to achieve a similar reduction in transmit rate.

2.7.2.3 Idle periods vs nofeedback timer

As previously explained, CCID3 makes use of a nofeedback timer to safeguard against events where loss of acknowledgment packets occurs. Although unlikely in CCID3 given the types of applications that make use of the protocol, there may be circumstances when the sender becomes idle and the time between acknowledgments exceeds the nofeedback timer. Instances where the no-feedback timer may be exceeded include conference calls where long idle periods exist which is a likely candidate for DCCP controlled streams. When this occurs, CCID3 will reduce its send rate by roughly halving the number of packets sent per RTT. In order to accommodate scenarios where the application has simply become idle and where there has not been acknowledgment of packet loss, CCID3 employs the techniques outlined in RFC3990 [38]. These techniques are used to ensure that there is a timely return to optimal transmit rates when the idle period ends.

To achieve this, RFC3990 specifies that the allowed sending rate is never to be reduced to less than the initial sending rate as the result of an idle period where at the time the idle period begins the allowed sending rate was greater than or equal to the initial sending rate [38]. This is irrespective of how long the idle period lasts. If however, the transmit rate was less than the initial starting transmit rate at the time
at which the idle period began, then there is no definitive specification in the RFC for how to handle this. It is therefore assumed that the halving of the transmit rate will be governed by the nofeedback timer specification stating that the allowed sending rate is never to be reduced to a rate below one packet per 64 seconds [8]. Once the application idle period ends, the sender can at most double its sending rate each RTT. This will continue until the sender reaches the allowed transmit rate determined by the loss event rate.

2.8 Conclusion

The second part of the chapter provided a summary of the various standards that make up the DCCP protocol and how each of these standards interact with one another. This summary forms the basis for the simulated standard DCCP model used in all subsequent chapters. A detailed summary of how the transmit rate is calculated for a CCID3 controlled ow is presented in the anatomy of a DCCP ow section to highlight the typical behavior found in a standard DCCP ow. The triggers that cause the transition from the slow start to congestion avoidance phase as well as the formulas used to calculate the transmission rate are also described to highlight the various phases each ow goes through.

In the next chapter the factors that affect the transmission rate are introduced. These factors have bearing on the transmission rate as they dictate the RTT values which the congestion control algorithms above utilize to calculate the transmission rate. Once all the factors have been introduced, Chapter 3 will explore specifically the impact queue size has on a DCCP ows performance.
Chapter 3

The effect of queue sizes on CCID3 performance

3.1 Introduction

This chapter explores the effects placing fixed size queues on transmission interfaces has on standard DCCP flows controlled by CCID3. In order to measure this, experimentation was carried out in Opnet Modeler using various simulated topologies that employed various fixed sized queues along the network path between the sender and the receiver. To deduce the effects enforcing these fixed size queues had on DCCP performance throughput, goodput, round trip time (RTT) and packet loss values were used. Following analysis of the results collected from the experimentation, discussion relating to the varying needs of applications and the correlating queuing strategies that should be employed for these needs is presented based on the findings. To conclude the chapter describes the scope and implications of the discoveries made through the research. In addition to the primary aim of measuring the effects fixed sized queues have on DCCP performance, the chapter also serves to introduce the simulated standard DCCP model that is used for the remainder of the thesis. The experimentation carried out provides validation that the simulated DCCP model created for this thesis functions in accordance with the various RFCs that make up the DCCP standard.
3.2 Background

One of the features of DCCP 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 such as RED [21], BLUE [22] etc. have been designed and since proven to be effective in ensuring optimal TCP 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 caused by the retransmission. This notion is however less acceptable in DCCP as dropping packets for the purposes of rate control results in those packets being lost permanently.

This would suggest that DCCP should be treated more in line with UDP which does not retransmit lost packets. However, UDP does not use loss events to reduce the transmit rate based on packet loss or network congestion. When a loss occurs due to congestion in UDP transfers, the rate is not reduced. 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, unsustainable levels of network congestion resulting from no congestion control in UDP seems inevitable.

Although DCCP is much simpler to understand conceptually 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. This chapter seeks to quantify the effects fixed queue sizes can have on DCCP and CCID3 performance over a single flow. By doing this the effects of rate control without lost packet retransmission are showcased.

3.3 Effects of queuing on CCID3 controlled DCCP flows

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. To determine how fast packets are being delivered, the RTT value is used. This value measures the total time that is taken for a packet to be sent and acknowledged across the network. As previously discussed, lower RTT values are indicative of faster delivery times of data packets. While increases in RTT values 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 path.

The server’s application send rate defines the number of segments the application layer passes down to the DCCP layer. If the application send rate exceeds the maximum achievable send rate governed by the bandwidth on the sending device then queuing on the outgoing transmission interface will occur. If the number of packets placed in the queue begins to increase while awaiting transmission, the RTT of packets will also begin to increase correspondingly. In addition to queuing taking place on the outgoing transmission interface of the sending device, queuing can also occur if there is a discrepancy in transmission rates along the network path. For example if packets are received on an intermediate network device such as a router at a rate that is faster than those packets can be serviced then queuing occurs.

Throughout a DCCP session queue depths fluctuate in size based on network contention. Such fluctuation causes variance in the delay experienced by packets traversing the network. Because the CCID3 congestion algorithm calculation is reactive to information based on previous network events such as last reported RTT values, this variation in delay can have a profound impact on the transmission rate. The experimentation in this chapter explored the effects prolonged RTT values have on the performance of CCID3 and specifically ways in which RTT caused by queuing of packets on transmit interfaces could be reduced in order to achieve the most desirable CCID3 transfer characteristics for real time applications.

As previously mentioned, if DCCP 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 no queuing of packets is required. In these circumstances the actual transmit rate is governed by the CCID3 transmit rate. This chapter only includes scenarios where the DCCP CCID3 transmit rate exceeded the interface transmission speed and queuing or dropping of packets on the transmitting interface was required. While the following experimentation only focuses on the transmit interface on the sending device, the effects queuing has on DCCP transmissions will apply to all transmit interfaces located on intermediate devices between the sender and the receiver.

3.4 Experiment aims

The experiments described in this chapter place hard limits on queue sizes on the network devices in order to determine if limiting the size of queues along the network path is able to prevent prolonged RTT values from occurring due to extended queuing periods. Large RTT values are undesirable to real time data due its finite lifespan. If packets are placed in queues for long periods the data they contain may exceed its useful lifespan causing the packets to become stale. While smaller RTT values are desirable, the act of placing hard limits on queue sizes means once a queue becomes full then packets arriving at the full queue are dropped. Therefore these experiments not only aimed to find ways of reducing RTT values for packets traversing the network but also ways in which this could be done whereby packet loss events did not negate the benefits of having packets with smaller RTT values. In order to accomplish this five measures of DCCP performance were measured: RTT values, number of packet losses, goodput, throughput and the packet loss ratio. Each of these measures of DCCP performance will be discussed further in the results section (Section 3.7).

3.5 Experiment design and methodology

To carry out the experimentation needed to determine the effect enforcing fixed size queues had on the performance of DCCP and CCID3, four simulated topologies were created in Opnet Modeler [6]. 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 four topologies employed a unique link speed for 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 run on each topology, with each iteration using a different maximum fixed queue size limit on the outgoing transmit interface on the sender. By doing this, the queue size was never allowed to exceed the predetermined amount. 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 to provide a range of simulations utilising various queue sizes.

This experimentation focused 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 impacted on the results. Additionally, it should be noted that while this experimentation relates specifically to the queue located on the transmitting interface on the sender, it is also applicable to any queue located between the sender and the receiver and therefore it is plausible that the results herein can be applied to the queues on intermediate devices such as Internet routers and switches.

### 3.6 Results

#### 3.6.1 Round trip times (RTT)

This section will explore the direct effects fixed queue sizes have on the RTT values. Table 3.1 shows the various RTT values that were recorded when different queue sizes were employed in the experiment topologies described in Section 3.5. The results presented in Table 3.1 are based on ideal network conditions where no loss or contention occurs during the simulated period.
From the results gathered from the experiments, it is clearly shown that employing smaller fixed queue sizes results in smaller RTT values as a result of the queuing process. The smallest fixed queue size of 10 packets resulted in the smallest RTT values in all four of the tested topologies. As the queue size limit increased so too did the average RTT value for the transmission. The RTT value is smaller when a small fixed queue size is employed because in small fixed size queues the compounding effect of the delay needed to service preceding packets in the queue is minimal when compared to that found in the larger fixed size queues.

When the transmit interface’s link speed (TX) is not able to service packets passed down from the higher layers quickly enough or in the case of an intermediate device, the packets arrive (RX) faster than they can be transmitted, then queuing on the outgoing interface occurs and the queue will begin to grow (i.e. $\text{If } RX > TX = \text{increase in queue size}$). 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 due to this compounding effect, there will be an increase in the RTT value and this will impact upon DCCP's performance as well as the performance of the application. In a queue with a fixed size of 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. From the results, the minimum delay time (MD) which a packet will be subjected to in a queue can be expressed as follows assuming that the queue is operating on a FIFO basis and that the link is able to be utilized at full capacity (Equation 3.1).

\[ MD = \frac{NP \times 8}{TX} \]  

(3.1)

Where:

MD = Minimum delay time a packet can expect in a queue.

NP = Number of packets existing in queue (in bytes).

TX= Transmission rate on interface (in bits/sec).

### 3.6.2 Packet loss

Before discussing packet loss and presenting results about packet loss 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, packet loss in this experimentation was recorded at two stages during the 120 second test duration. The first set of results was gathered at 50 seconds (slow phase has ended in all experiments) and the second set of results was then collected at the end of the 120 second experiment session.

Table 3.2 shows the results collected after 50 seconds and 120 seconds for the DCCP CCID3 flow relating to the packet loss experienced when different fixed size queue and network link speeds are used.
Table 3.2: Packet loss rates experienced during experimentation.

From the results, it becomes apparent that small fixed size queues (10 packets) and large fixed size queues (200 packets) result in the largest number of packet drops compared to fixed queue sizes of 20, 50 and 100 at the tested speeds. Deeper analysis however shows that once the slow phase has ended, the larger fixed queue size tested (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 a fixed queue size of 200 occur due to the Slow Start and subsequent termination of the Slow Start phase. Once this phase has completed, the number of dropped packets becomes negligible and drops occur at much more infrequent intervals.

In bigger fixed size queues the number of packet drops does not occur as frequently 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 in the large fixed size queues and therefore the algorithm is less reactant to packet
loss. This is due to there being a much longer period available for acknowledgments of packets compared to smaller fixed size queues where the average RTT is much smaller and the congestion algorithm more reactant. An increase in the maximum queue size results in a smaller number of packet drops occurring in the period after the initial 50 seconds.

On the other hand the smaller fixed size queues experience continual packet drops throughout the life of the session as the queue’s maximum level is reached continually throughout the transfer. The results show 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. Support for this is provided 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.

With the exception of the 96 kbps topology, the use of a larger fixed queue size results in a smaller number of packet losses after the initial Slow Start phase is complete when compared to smaller fixed size queues. The anomaly shown in the 96 kbps topology was found to be caused by the fact that 75 packets were lost in the slow start to equation based flow control transition. Within the 120 second period, the slow 96kbps link did not allow a sufficient number of packets to traverse the network to allow the small queue size packet loss account to grow larger than the 75 packets loss that the 200 packet fixed size queue experiences as a result of the Slow Start phase. The experiment duration was later increased to 10 minutes and the results showed that a fixed 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. The fixed queue size of 200 resulted in a total of 82 packet losses after 10 minutes compared to the 816 packets lost for the queue size of 10 over the same period utilizing the 96 kbps topology. This supports the proposition that a larger fixed queue size will result in fewer packet loss events when compared to smaller fixed queue sizes.

3.6.3 Throughput and goodput

From these results, there is clear evidence to suggest that larger fixed queue sizes result in fewer packet drops over sustained periods. However, employing larger fixed 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 terms of loss and low RTT values. In an attempt to add further clarity to this tradeoff, an additional group of results relating to the transfers will now be presented.

Throughput is defined as 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 loss ratio will be used to represent the number of lost packets as a fraction of the total number of packets sent.

Table 3.3 shows the results pertaining to throughput, goodput and the loss ratio experienced after 50 seconds and 120 seconds of the DCCP CCID3 flow.

![Table 3.3: Goodput, throughput and loss ratio values recorded from the topologies using various fixed size queue settings.](image)

By expressing the results in terms of throughput, goodput and loss ratios, it once again becomes apparent that there is a definitive tradeoff between having faster transfers and more reliable transfers as governed by employing fixed queue sizes. In all cases where the smallest fixed 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.

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

The results do suggest that a moderate fixed queue size of 50 packets results in the best packet loss ratio for 120 seconds in all the experiment topologies except where a link speed of 2048kbps was employed. In this case the slightly larger 100 packet fixed sized queue was the best performer. This was only marginally better than the queue size of 50 and was deemed to be an anomaly rather than the optimal queue size.

3.7 Discussion

3.7.1 Various forms of real time applications

The results demonstrate that through the utilization of smaller fixed 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 potential applications that require real time data transfer, however the effects 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 in the experimentation and seeks very small RTT times then small fixed queue sizes should 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 for situations where high loss is less acceptable to the application will now be proposed.

**Empowering the application approach**

The first approach involves empowering the application by making it responsible for packet retransmission. The results suggest that in order to achieve optimal RTT values fixed queue sizes should be made small which will lead to a higher level of packet loss. To compensate for this higher volume of loss this approach recommends the application should be made to perform retransmissions of certain lost packets. If the application layer were to be made responsible for the retransmission of important lost packets then it would allow the application the ability to determine whether or not a particular packet should be retransmitted. For example in a H.264 [39] video transmission 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 negated. This scheme is only possible if the RTT is small enough to allow the retransmission to occur within the useful lifespan of the data.

**The “happy-medium” approach**

Another solution that can be used by the protocol is to select a “happy-medium” approach whereby a medium (relative) fixed sized queue is selected on the outgoing interface that offers neither the best nor the worst performance in terms of loss or RTT. In the experimentation, it was shown that fixed 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
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 fixed queue sizes on outgoing transmit interfaces becomes problematic. Where multiple streams existed 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 to two or more applications with different requirements would be extremely difficult and computationally exhaustive.

**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 Chapter 6 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 are then 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 the 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 would be able to maintain a single queue but offer the best performance to all applications utilizing an interface through various scheduling techniques.
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 if larger fixed size queues were utilized. 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 that early exiting of the Slow Start mechanism would have on CCID3 performance.

3.8 Limitations of this experimentation

The simplicity of the experimentation described in the chapter was deliberate in order to test the impact fixed size queues could 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 RFC Standards. Secondly the experiment aimed to generate a baseline of the results that DCCP is capable of producing at in ideal situations. While these objectives have been met, the experimentation's design elements and scope must be taken into consideration when using the findings presented above. In this section some pertinent elements of the experiment will now be discussed for the purposes of rigor and recommending future research.

3.8.1 Application send rate

The RTP layer used in this experimentation was designed in such a way 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 likely act in such a
manner. Instead a steady stream of varied sized 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 G.711 codec [37] would send a varied stream of approximately 64 Kilobits/second across any links capable of servicing this rate and provided there was no other contention for the bandwidth irrespective of the link’s speed. This would equate to an average around 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 of worst case, whereby it is used in environments where high levels of congestion are being experienced on networks and where queuing is taking place due to a disparity in received packet 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. It was deemed reasonable in this research to use elastic DCCP flows as an initial test case for the proposed modifications in this thesis, however it is inappropriate to claim that flows are representative of multimedia applications.

3.8.2 Effects of packet drops on the application

In addition to the application not controlling the send rate through a fixed rate, this experimentation did not explore the impact lost packets have on the application specifically. The reason for not entering this area of research is simple. There are so many 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 able to become responsible for the potential to control 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 the direction of self governed retransmission.

3.8.3 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. These phenomena will be explored further in Chapter 5 of this thesis. In addition the network does not take into account normal incidences such as unexpected packet drop, fluctuations in delay and congestion or the use different route paths for a single flow as they were deemed out of the scope of the experiment objectives.

3.9 Conclusion

This chapter explored the effects utilizing fixed sized queue sizes has on DCCP performance when CCID3 is used to provide congestion control. The results presented in this chapter have shown that employing small fixed size transmit queues lessens the RTT value experienced by packets traversing the network but leads to a higher and more sustained packet loss rate. Unlike TCP which retransmits lost packets, these lost packets are not sent again 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 and queue size determination. 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 that is able to take control of queue sizes and depths for multiple simultaneous streams from different applications
is alluded to. In addition to demonstrating the effects fixed queue sizes have on RTT values and ultimately DCCP and CCID3 performance, this chapter has also served to introduce and verify that the simulated implementations of DCCP, CCID3 and RTP are functioning correctly and operating in accordance with the various RFCs.

Finally, through the experimentation described above, this chapter has also produced a collection of baseline DCCP performance data to which later comparisons can be made as new contributions are added to the various DCCP standards.
Chapter 4

Incorporating deadline scheduling into DCCP

4.1 Introduction

This chapter comprises of three distinct parts. In the first part of the chapter, the theory of incorporating a deadline scheduling scheme into DCCP is presented. In addition to describing how deadline scheduling is incorporated into DCCP, the benefits and possible uses for the deadline scheduling scheme in DCCP are also discussed from a theoretical perspective. In order to test the feasibility and effectiveness of the scheme proof of concept modeling is carried out.

In the second part of the chapter, the experimental model used to carry out this proof of concept modeling of the deadline scheduling scheme described in the theory section is presented. This section adds to the theoretical component by discussing how the actual implementation was carried out as well as issues that were experienced and how these issues were overcome.

In the third part of the chapter, the results of experimentation carried out on the proof of concept model are described and analyzed. While the results gathered from the experiments proved that the scheme is feasible and stable, they also explored the effects the additional overheads needed to implement the scheme had on DCCP performance. It was found that the additional header size needed to incorporate packet life variables into the DCCP standard did reduce the overall efficiency of the DCCP protocol marginally. The extent of this detriment is quantified through a
series of experiments on DCCP flows comparing the modified variant of DCCP that incorporates deadline scheduling to a standard DCCP implementation.

4.2 Motivation and background

One of the biggest reasons for the creation of DCCP is that TCP and UDP are not ideally suited to handle the needs of modern real time traffic [3]. TCP is too overhead intensive and aggressive in congestion avoidance, whereas UDP offers no congestion control management. While DCCP provides a best of both worlds approach, it still fails to take into account the finite life span of the data it is transporting. As DCCP (using CCID3) is primarily designed for the transfer of time sensitive real time data, it is curious that DCCP does not support or incorporate some of the existing techniques that have been designed for real time applications in other transport protocols such as UDP or some of the theoretical mechanisms proposed in research such as [20, 15, 16, 26, 13, 40]. The reasons for this are most likely attributable to the relative infancy of the protocol and the fact that no research has been carried out to find out if in fact these mechanisms would provide any benefit to the protocol if incorporated. In order to address this deficiency, research was undertaken to determine how deadline scheduling can be added to the current DCCP protocol standard.

Specifically, a modified DCCP protocol that incorporates packet life information through the addition of a deadline scheduling scheme is presented to offer an alternative approach to the process of transferring real time data. In current networks including DCCP networks, every packet is essentially treated as the same one dimensional entity regardless of the data it is carrying (exceptions include QoS schemes such as class based weighted fair queuing (CBWFQ)). In reality, the data the packet is carrying could potentially be stale or old rendering it useless to transmit and receive. Regardless of the fact that it has expired, in current networks, a packet will continue to traverse the network until it reaches the endpoint in the same way a packet that has not expired will traverse the network. This may be to the detriment of packets that have not yet expired, but expire due to the additional time needed to send the stale packet.

Instead of treating every packet the same and essentially offering best effort based
delivery to every packet, the deadline scheduling enabled DCCP protocol described in this chapter adopts the notion that not all packets are necessarily equal and/or the same. By obtaining information from the application and embedding it into the DCCP packet header, a second dimension is added to the information known about the packet that is not currently available to the transport layer.

By using this information relating to the lifespan and age of the data, the transport layer at intermediate points in the network can make more informed decisions about how best to schedule packets in their queues. In addition this approach empowers the application by allowing it the ability to provide information regarding the characteristics of the data it is trying to send to the transport layer which again is not currently possible in DCCP. While this information is treated only as a recommendation and not necessarily a requirement, the role of the transport layer now becomes one of trying to offer a more tailored level of service to the application layer rather than the generic one size fits all model that currently exists. Even though the application layer is given some influence on the transport layer, the scheduling techniques used on intermediate points (BODS) are designed to offer the best service for the greatest good to all packets. The foundation roots for the notion of greatest good for all through proper sequencing of tasks and/or events based on gathering more information in order to yield incremental improvement to the entire group stems back to [41].

Specifically the research presented in this chapter incorporates deadline scheduling into the DCCP protocol to allow the application layer to pass packet life information down to the transport layer (DCCP) in order to allow more efficient scheduling of real time data packets. In the version of DCCP presented in this chapter DCCP endpoint devices embed packet life information passed down from the application layer to the DCCP layer into DCCP packet headers. In order to add the facilities needed to carry out deadline scheduling a number of minor modifications were made to the DCCP protocol. These included the addition of a maximum lifespan time to live (MTTL) variable in the DCCP protocol header as well as having the application layer pass lifespan information down to the transport layer through an appropriate API to populate this field. This chapter will define both of these additions in accordance with the DCCP standard as well as all other changes required for deadline scheduling to work. Note this chapter only describes changes to the DCCP protocol at the endpoint
devices. Intermediate devices and how they respond to the packet life information placed within the DCCP header will be described in subsequent chapters (Chapters 5 and 6).

4.2.1 Deadline scheduling in DCCP

As explained in Chapter 2, deadline scheduling in the context of this thesis refers to the process by which packets are transported, prioritized or dropped based on the packet’s remaining useful lifespan. In this scheme, the useful lifespan is conveyed to the transport layer by the application layer and specifies how long the data can exist for before becoming stale, thus giving the application an element of control as to how the transport layer should handle the data.

No research offering deadline scheduling specifically in DCCP environments has previously been attempted. In addition, most of the previous research in deadline scheduling, examples of which can be found in [40, 20, 15, 12, 29], is of a purely mathematical nature and does not delve into the actual implementation of the described schemes. In one of the few studies carried out on a simulated representation of an actual network, Wong and Liu [19], adapt the theoretical principles described in [41], and implement their own deadline based algorithm variation in a simulated IP based network utilizing TCP at the transport layer. Wong and Liu claim to be the first to embed a finite time to live value into an IP packet header based on information passed down from the application layer. The results gathered from their experiments are very positive and show deadline scheduling to work more efficiently than simply using a FIFO scheduling technique. However, one of the biggest problems with the approach described in [19] is that the scheduling technique they employ, which prioritizes packets in an outgoing queue based purely on the packet’s reported time left to live value, creates the potential for a number of fairness related issues. In particular, the simplistic scheduling mechanism introduces potential for miscreant applications on a network to set their time to live variable to unreasonably small values and in doing so would allow the application to seize an unfair portion of the available bandwidth. The scheme also introduces the potential for denial of service (DOS) type attacks through this same vulnerability. If miscreant hosts generated large quantities
of packets with very small time to live values they could prevent other packets from other flows with larger time to live values from ever being delivered.

In [20], Figueira describes a model in which the application does not place the deadline into the packet header but instead this task is left to intermediate devices to mark a packet’s maximum lifespan. While this alleviates the possibility of misbehaving endpoints, this scheme offers no scalability as intermediate devices would need to be aware of all the application specific requirements prior to becoming functional. As a result this scheme would not scale well in uncontrolled networks such as the Internet where exact traffic types are unknown.

As allowing applications to inject their requirements into network packets is far more scalable, the research described in this chapter will investigate how such a scheme could be incorporated into the DCCP protocol in a safe and secure manner whereby miscreant applications are not afforded any benefit from generating packets with unreasonably small time to live values. It is important to note that the process of an application passing the maximum TTL variable down to the transport layer does not in itself pose any security concerns to the protocol. Where security concerns such as those found in [19], are introduced is when the scheduling mechanism on an intermediate device makes scheduling decisions based on this information in an unsafe manner: for example in [19], where packets with a lower time to live are given priority over packets with more time to live.

In order to reduce the risk potential for misbehaving nodes on networks two unique schemes which can be used individually or in conjunction will be introduced in Chapters 5 and 6. The first scheme described in Chapter 5 discards packets which have lived beyond their useful lifespan. This means if a misbehaving node sets its maximum time to live variable to a value too low then the packet may expire before reaching the intermediate device and will be discarded as the data will be deemed stale. Note while this scheme is designed to optimize DCCP transfers and not specifically for the role of preventing misbehaving nodes, it will provide protection mechanisms against such devices.

The second scheme described in Chapter 6 and known as probability based scheduling provides a much greater deterrent for misbehaving nodes. In this scheme, a novel scheduling algorithm is introduced to calculate the probability a packet has of arriving at its intended destination within its useful lifespan. Scheduling of packet delivery in
a queue is then based on this probability. This mechanism ensures there is no benefit in a misbehaving node setting their maximum time to live to a lower value as this would mean there is less probability of that packet arriving in time and result in that packet being given a lower priority. The goal is to achieve an environment where applications set their maximum time to live to an ambitious but at the same time realistic value. Chapter 5 and 6 describes these processes and their merits in greater detail. This chapter aims to describe the deadline scheduling capable DCCP protocol which makes such schemes as those found in Chapter 5 and 6 possible.

4.2.2 How the deadline scheduling mechanism works

As with most literature describing network communications, it is important to note that communication between two DCCP hosts is bidirectional. As such, during communication it is common for two unique connections known as half connections to be established (i.e. one half connection for each of the directions). Each half connection comprises of a sender/server and a receiver/client. Although they are logically distinct, in practice the half connections often overlap [4]. Specifically, the DCCP standard allows acknowledgments to “piggyback” on data packets by using the DCCP-DataAck packet type. As a result a DCCP-DataAck packet could potentially contain application data relevant to one half connection and acknowledgment information relevant to the other half connection [4]. For the purposes of simplicity during this narrative description, only one half-connection will be described when detailing how deadline scheduling is incorporated into the standard DCCP protocol.

4.2.2.1 High level overview

The session is started when a client sends a request to receive data from a server. Examples of this would be clients wishing to access stock market data or watch a live video stream. After a process similar to TCP’s three-way hand shake has occurred between the client and the server the DCCP connection is established. At this point the application layer on the server passes the requested data down to the transport layer (DCCP). At the same time the application specifies the maximum time to live
(TTL) value for the real time data it is transferring and passes this variable to the DCCP protocol. In the case of a Voice over IP call using the G.711 codec [37], the application could determine that data should be received within 200ms. Therefore it will pass the value of 200ms down to the DCCP process with the data.

Upon receiving the data and the maximum TTL value for that data, the DCCP protocol encapsulates the data into a DCCP-Data packet. This includes a generic DCCP header followed by two custom option fields and finally the payload data. The two custom option fields are used to store the maximum TTL variable passed down from the application layer as well as the exact time the DCCP segment is created (known herein as the packet’s “birth time”). This birth time is derived from the NTP protocol and specifies the time at which the segment is encapsulated into a DCCP packet. These values provide a reference for calculating how long a packet has existed for and how much longer it has remaining before it expires.

In order to keep the overheads of the headers smaller, it would be possible to simply take the exact NTP time and add the maximum TTL variable to it in order to come up with an explicit expiry time, however it was felt maintaining the maximum TTL variable allowed for more flexibility and provides more information to the scheduling algorithm on intermediate devices that make use of this information. Using only an explicit expiry time would mean the scheduling algorithm would not be able to determine if the packet had expired due to an unrealistic maximum TTL variable being placed on the packet or if it had been delayed prior to arriving at the BOD.

Having embedded the maximum TTL value as well as the birth time into the packet header, the DCCP data packet is transferred to the appropriate network layer protocol and then is sent across the network. Intermediate devices in the network between the sender and the receiver can then use this information to determine which is the most effective means of scheduling packet delivery on the network. Upon receiving the packet, the end point or receiver will simply ignore the additional options fields. The packet will still be decapsulated and the data passed to the relevant application layer above. Having provided a high level overview of how deadline scheduling is incorporated into DCCP, the following sections will now delve into the specifics that need to be implemented in order for this to occur.
4.2.2.2 NTP synchronization

This scheme requires all devices in the topology to use and be synchronized with the network time protocol (NTP). Desynchronisation of end point devices utilising the mechanism through clock skewing and timing variations would have a major destabilising effect on the overall protocol. While the exact workings of NTP are beyond the scope of this research, NTPv3 and the less widely adopted NTPv4 are able to provide millisecond and beyond accuracy. In [42] a comprehensive study is carried out to determine the availability of NTP and increase in NTP adoption on the Internet. The research shows that the NTP network size grew from 38,722 hosts in 1995 to 647,041 in 1999 to 1,290,819 hosts in 2004 [42]. This shows that the choice of NTP for the purposes of deadline scheduling is feasible and that more services are becoming dependent on NTP. In addition the research also examined the quality of NTP time services that are available on the Internet. The results showed that there has been a steady increase in the accuracy that can be found on NTP networks since earlier experiments were carried out in 1995. During their experimentation, the authors in [42] found the offset between NTP devices to have a median of 0.7ms and a mean of 7ms. This shows that NTP is capable of providing accurate enough timing information for the purposes of this thesis. It is also hypothesized that the offset in current times would be far less than those occurring in 1999 due to the exponential adoption on NTP found in [42], however there is no definitive research to support this hypothesis conclusively. Additionally it must be noted that the above times are provided to show what NTP performance is possible via the Internet NTP infrastructure. If a NTP topology was deployed within a LAN environment, the offset between devices would be much less. In [43], the authors demonstrate a GPS NTP based time synchronization technique that is capable of obtaining as little as 0.4ms offset between the clocks on two devices.

Having shown that NTP is both well enough supported and accurate enough to satisfy the needs of incorporating deadline scheduling into DCCP, it is hoped the reliance on it does not overshadow the benefits that can be obtained from the scheme. If the use of NTP is not possible, then there is scope for other synchronization techniques such as those found in [44, 45, 46] to be used. These are however beyond the scope of this research and will therefore not be discussed.
The implementation of the deadline Scheduling scheme in DCCP will require that NTP be enabled on all DCCP devices and that the system clocks be synchronized to the same higher strata NTP device. The DCCP protocol itself will not make any checks to ensure this has occurred and will assume this step has been taken. The DCCP deadline protocol will simply query the NTP time protocol or the system clock on the device when required and as necessary for the given operating system/firmware. How the NTP time value will be embedded into the DCCP header will now be outlined.

4.2.2.3 DCCP header modification

The DCCP header will not be modified as such to accommodate the deadline scheduling scheme, instead a set of variables relating to packet scheduling will be appended to the end of the DCCP header in the form of options. This ensures the DCCP header remains intact according to the current RFC standard outlined in [4]. In addition this also allows non deadline scheduling enabled devices to still use the DCCP packet as they will simply ignore the option types they do not understand as dictated by finite state machine used to implement DCCP [4].

Initially it was thought that the easiest way to implement the scheme would be to simply modify the DCCP header and add the necessary fields. The problem with this approach however is twofold. Firstly standard implementations of DCCP would not be capable of processing the new headers and as a result the entire standard would need to be changed to accommodate deadline scheduling. Secondly, modifying the generic header on all DCCP packets is unnecessary as deadline scheduling information is only required on data carrying packets (DCCP-Data packets). Including the additional fields in all other types of DCCP packets by changing the generic DCCP header found in all packets would add unnecessary overhead and size to non DCCP-Data packets and contravene DCCP’s minimalistic ethos. As such it was deemed that the generic header should remain untouched and instead the above mentioned options be appended after the generic DCCP header for DCCP-Data packets. Figure 4.1 shows diagrammatically the generic DCCP header. The two additional options fields needed to incorporate deadline scheduling into DCCP are placed into the options and padding portion of the packet.
Another idea considered before settling on the choice to use options was to make use of the maximum segment lifetime (MSL) variable found in the generic header in DCCP. This value is used to ensure data segments do not exist in a network for excessively long periods. Given this it was initially thought that this was an ideal location to place the application specific maximum TTL value which by default is set to 120 seconds. However, after exploring the uses for the MSL value this is option was deemed unsafe as number of key congestion control and error checking mechanisms rely heavily on the value. Allowing the application to specify arbitrary values, especially very large and very small MSL values, would cause the protocol to become unstable. If, for example, the value was set to a very small number, connection resets would occur frequently if the data was not delivered within the small time frame. This would cause the connection to reset and open the protocol up to DOS type attacks. While there is much need for research investigating the ideal MSL values in DCCP, this is beyond the scope of this thesis. Given the dependence on the MSL value, it was concluded a separate option field should be created for the Maximum TTL value in order mitigate the potential for applications to cause the DCCP stack to become unstable.

4.2.2.4 Option fields

As mentioned previously, in order to make an endpoint device deadline scheduling compliant, two additional option fields are required in all DCCP-Data segments. These fields carries the birth time variable as well as the maximum TTL variable.
If the endpoints were to follow a predetermined path, it would be possible for the maximum TTL variable to be transferred only once per connection. However given the distributed nature of IP networks, the path may change throughout a session and as a result each segment is therefore required to carry this value as intermediate devices that will act upon these values may not be present in the path if a once off specification of the maximum TTL value was to take place.

When creating the options, the DCCP standard dictates that all options must possess a size value which is a multiple of 8 bits and the cumulative total (in bits) of all options must be a multiple of 32 [4]. In instances where the size of the options is not a multiple of 32, padding is to be used.

The first option that is appended to the header is used to embed the DCCP segments birth time timestamp into the data segment and is given the type number 51. The second option is used to embed the maximum lifespan TTL value into the header. The exact type number will need to be allocated and approved by the IETF task force currently working on the DCCP standard. For the purposes of this thesis however, types 51 and 52 will be used for the proof of concept modeling.

Figure 4.2 shows the structure for the NTP birth time timestamp option. This option provides an exact record of the time at which the DCCP header was added to the segment. This value comprises of a 64 bit timestamp value that is derived from either the NTP protocol directly or the operating system/ firmware clock depending on the implementation. As in NTP, 32 of the 64 bits will be used for seconds while the other 32 bits are used for fractions of seconds.

![Option 51: NTP Timestamp](image)

Figure 4.2: Additional NTP option field for deadline scheduling scheme.

The choice of a 64 bit value was made to ensure the protocol remains consistent with the NTP standard [45]. There is no reason why a more recent epoch than 1990
could not be chosen (as currently found in NTPv3) and a smaller variable bit size be used, however, this would require constant calculation on the part of the DCCP process to determine the relative time. It is simpler computationally to use the NTP value even though it introduces slightly higher overheads in terms of packet size. Additionally this will avoid frequent roll-over events which could potentially cause instability in the network.

Figure 4.3 shows the structure of the maximum TTL option field utilized for this scheme. This value is passed down to the DCCP layer by the application layer simultaneously with the passing down of the data needed to be transmitted. This value is given in milliseconds and represents the amount of time the data can exist for before it becomes stale.

![Option 52: Maximum Lifespan Time to Live](image)

Figure 4.3: Additional maximum TTL option field for deadline scheduling scheme.

4.2.3 Non compliant devices

The use of options rather than DCCP header modification allows backward compatibility for non deadline scheduling enabled devices. The DCCP specification states that devices MUST ignore options they are not aware of [4]. For this reason, end point devices that are non deadline scheduling enabled will treat the segments as per normal. In the case of the receiver, the segment will be decapsulated and passed up to the application layer. As IP will be used at the network layer, most intermediate devices such as routers will only function up to the IP layer and therefore are transport layer protocol independent. DCCP segments will traverse IP devices such as routers unabated. Only if manipulation is required will the intermediate device need to be DCCP scheduling enabled.
4.2.4 Possible uses of the newly proposed option fields

Intermediate BOD devices located on the network path between the sender and the receiver will likely be the main devices making use of the values stored in the options above. In Chapter 5, five techniques for the DCCP protocol will be introduced in which packets that have existed beyond their useful lifespan as advertised by the maximum TTL value are deleted/purged as they traverse through intermediate BODs. In Chapter 6 a novel probability scheduling scheme will be described that will take advantage of the options made available through the incorporation of deadline scheduling in DCCP. In the scheme presented in Chapter 6, packets will be ranked based on their probability of arriving at their intended destination within their useful lifespan in order to optimize data scheduling on intermediate devices. To determine the probability a packet has of arriving at its intended destination, routing protocol information is used.

4.3 Experimentation

This section will describe how the DCCP protocol with deadline scheduling was modeled for the experimentation carried out in this chapter as well as subsequent chapters in this thesis. This section will only describe how end point devices were made deadline scheduling ready/capable. For the purposes of this chapter all intermediate devices in the network topology are standard store and forward devices that utilize a first in first out (FIFO) scheduling technique. Following the narrative on how the modification process of DCCP was carried out, experimentation to show that the model is in compliance with the DCCP standard will then be presented. The results from this experimentation validate that the modified DCCP variant reacts in accordance with DCCP standards and secondly quantify how much detriment the additional overheads of the options needed for the deadline scheduling functionality have on DCCP performance.

The modeling used for all simulations in this research was performed in Opnet Modeler [6]. At the time of this research Opnet Modeler did not have a standard library for the DCCP protocol so an existing model found in [11] was imported into the application and used. After importing the protocol found in [11] into Opnet,
rigorous checking was carried out and modifications were made to ensure the model was compliant with RFC4340 [4], RFC4341 [7], RFC4342 [8] and RFC5762 [10]. Once this stage was complete, the task of making the DCCP protocol deadline scheduling enabled commenced.

4.3.1 Simulation toolkit

The research described in this chapter shows how deadline scheduling is incorporated into DCCP and then tests the stability and efficiency of this enhanced DCCP protocol in a number of different scenarios using simulated network topologies. The use of a simulation toolkit allowed for some liberties to be taken in order to simplify and speed up the implementation of the deadline scheduling scheme. Liberties refer to actions that can occur in a simulator but would not be possible in a real world network such as the use of global variable stores and a single simulation clock value. It is imperative to mention that these so called liberties were only taken to reduce the implementation time of the scheme and were only done if they would result in exactly the same performance that would be expected if the scheme was implemented in the real world. In order to offer fully a transparent overview of what was done the following sections (Section 4.3.1.1 and 4.3.1.2) will now provide an overview of the liberties taken.

4.3.1.1 Timing and NTP

As the simulation was carried out in a controlled simulation environment, it was deemed unnecessary to add an NTP infrastructure into the topology. Instead, during the experimentation all time variables were drawn from the global simulation clock in Opnet through the `op_time()` and `op_get_time()` functions. This technique resulted in an almost identical mechanism to using a NTP infrastructure. Instead of querying an NTP server for the time, the simulation clock in Opnet was queried. The advantage of this technique is that it ruled out the possibility that variation in results was caused by NTP desynchronization. While results in real world topologies may vary slightly to the ones presented here due to the possibility of the potential for desynchronization
between NTP nodes, if NTP is deployed correctly, this variation should be negligible.

4.3.1.2 Application layer design for simulated model

As the experimentation was carried out in a simulated environment, an application layer that replicated the real world equivalent of an installed application on the end-point devices was required. The choice of which real time application was to be used was not of major consequence given the fact that only the maximum TTL and birth time were used and that the scheme is independent of a specific application protocol. However, in the interests of making the research as applicable as possible to what would likely be found in a real world network, a Voice over IP (VoIP) application was created to generate the real time data.

Initially an application layer was developed to create a standard G.711 [37] voice call stream and was configured to pass data down to the DCCP layer. Specifically the behavior of a voice call utilizing the RTP [47] protocol in combination with the Pulse Code Modulation (G.711)[37], codec was modeled. The G.711 codec creates a 64kbit lossless audio channel and utilizes a payload size of 160 bytes [48]. To maintain a 64kbit rate, the codec requires a send rate of 50 packets per seconds (\[\text{codecbitrate} = \frac{\text{payloadsize}}{160}\text{bytes}\]). In addition to the 160 byte data payload, the additional overheads for the RTP protocol also needed to be included in the modeling. The RTP header requires 12 bytes (if compress-RTP is not enabled).

In order to create a stream of packets indicative of an elastic RTP and G.711 call between a sender a receiver, the application layer modeled was configured to pass data down to the transport layer at a rate of 50 segments per second or 1 segment per 20ms and 172 bytes per segment. To perform this function a customized packet was created in Opnet of 172 bytes containing arbitrary data. The application layer was then programmed to send one of these packets down to the DCCP layer every 20ms for the duration of the simulation to generate the desired elastic flow of packets.

The simulated application does not model connection establishment or tear down and therefore the modeling was only indicative of a data flow once connection has been established. At the receiver, the packets passed to the application layer were sent to a data sink upon being received in order to free memory resources in the simulator. Addressing information required at the RTP, TCP and IP layers was
configured statically. Additionally link fragmentation and interleaving was turned off on all devices. These settings are standard for all experiments in this thesis unless explicitly defined as being otherwise.

After carrying out a number of trials it was determined that the performance of the DCCP protocol became a function of the frequency at which the packets were passed down from the application layer rather than a product of the DCCP performance. During data transfer, distinct periods appeared where the DCCP layer was idle whilst waiting for packets to arrive from the application layer. While this is typical behavior, this phenomenon limited the likelihood of the results showcasing the maximum potential benefit possible by the scheme. Subsequently, the application layer was altered from sending one G.711 segment every 20ms to a mechanism where a G.711 packet was continuously passed down to the DCCP layer by the application layer. To achieve this the application code was modified so that the event of passing a segment down to the DCCP layer triggered the creation and queuing of the next G.711 packet at the application layer. This segment was queued until the DCCP layer was ready to handle it. By doing this the DCCP layer was always active and all performance data could be attributed to the DCCP protocol mechanisms rather than the mechanisms and timeliness of the application layer.

4.3.2 Network topology

To ensure the deadline scheduling enabled endpoints functioned correctly two network topologies were created to compare the performance metrics of the modified deadline scheduling enabled DCCP endpoint to a standard DCCP endpoint. Once created, two specific aspects of DCCP performance were tested. The first aspect tested was to ensure non-compliant devices continued to forward the DCCP segments containing deadline scheduling options across the network in a stable manner. Secondly the experiments aimed to ensure the performance of the modified variant of DCCP (deadline scheduling enabled DCCP) was somewhat similar to the performance found in standard DCCP implementations. While it was expected that there would be some slight difference due to the slightly larger header used to store the two additional options fields in the modified variant of DCCP, this addition was not expected to cause
major detriment to the protocol. The experiments carried out as described in Section 4.4 quantify the extent of this detriment.

The network topologies used in the experimentation are depicted in Figure 4.4. In the first topology a standard DCCP endpoint sender and receiver were placed on either end of the network. Located between them was an Internet cloud and four WAN links. The function of the IP cloud was to provide routing facilities and to add statically defined network load, delay and packet loss to the network as and when was required. The WAN links could be configured to provide any desired propagation speed and these speeds were changed during the experimentation to compare DCCP performance at various speeds. The router function provided by the IP cloud was identical to that of a standard layer 3 (IP) router. All IP addresses were statically assigned and static route entries were placed onto the IP cloud. There was no routing protocol enabled to prevent routing table updates from affecting the results by adding to the congestion that was created in the network. The router function on the IP cloud was also configured as a store and forward device with statically defined queue sizes. Congestion in the network was simulated by introducing random latency into the network. To achieve this pseudo-random latency values generated through an Erlang distribution curve were applied to the intermediate routing device found in the Internet cloud. This was done to inject random delay into the network to simulate congestion. Erlang may not necessarily represent exact congestion events but does reflect varying delay and programmatically was the most accurate random delay that could be implemented in the simulation. In addition to the Internet cloud, there were also two BODS placed between the DCCP devices. These are utilized in the research described in Chapters 5 and 6. During this experimentation, DCCP functionality was disabled on the BODs and the devices functioned only as high speed layer 2 switches.
The second topology was used to test the modified deadline scheduling enabled DCCP endpoints and consisted of exactly the same devices found in the first topology with the exception of the endpoint sender and the receiver which were modified to incorporate the deadline scheduling capability. All other characteristics in the network topologies were identical and any variation in results could only be attributed to the additional option fields appended to the header of packets transmitted by the deadline scheduling enabled DCCP endpoints.
4.4 Experiments testing the effects of incorporating deadline scheduling into DCCP Networks

4.4.1 Experiment aims

One of the design goals behind the creation of DCCP was to develop a protocol that utilized minimal overheads in order to keep packet size as small as possible. The modification to DCCP described in this chapter which seeks to add an additional 16 bytes to DCCP packets contradicts this design principal. Before subsequent chapters will demonstrate the potential benefits that can be achieved through the deadline scheduling scheme, it was felt necessary to first quantify the extent of detriment and/or performance degradation, if any, was caused by the the additional overheads needed. To achieve this, six key variables are measured through the experimentation. These include, the rate of CCID3 send window growth in both highly congested networks and non-congested networks, the effects larger packets had on CCID3’s RTT values, goodput and throughput, and finally CCID3’s reaction to packet loss events during an established session.

4.4.2 Methods of results collection

The simulated topology used to carry out the experimentation is described in detail in Section 4.3.2. Each simulated topology used either the standard DCCP model or the deadline scheduling enabled DCCP model while all other network characteristic remained identical. In doing this, variations in the results could then be directly attributed to changes in the way in which DCCP operated. To obtain a wide spectrum of results, a range of network speeds were used for the testing. For these experiments, four network link speeds were employed ranging from a slow 96 kilobit per second (kbps) link to a faster T3 link (operating at 44736kbps). Every experiment was performed four times with each iteration using one of the four network link speeds. The queue size was set to 100 packets for all experiments.
4.4.3 Experiment 1: Detriment to window growth rates

In these experiments, the send rate window size experienced by a standard DCCP model was compared to the send rate window size experienced by a modified deadline scheduling enable DCCP model. This was done to determine the level of detriment the additional options needed for deadline scheduling would have on the time it took the CCID3 protocol to stabilize. In addition to this, the eventual rate where the protocol stabilized was also measured for comparative purposes. To ensure robustness in the results, two varying sets of network conditions were employed in the topologies used to carry out the experimentation. In the first topology type, no congestion or contention for network resources occurred anywhere on the network path between the sender and the receiver other than the data generated from the testing flow.

In the second topology type, congestion was introduced into the network. In order to mimic congestion in the simulation, pseudo random delay values were injected into the internet router located between the sender and the receiver. The pseudo random latency values were derived from within an Erlang distribution curve configured with a scale of 0.0125 seconds and a shape of 2 for the 1024kbps, 5006kbps and 44736kbps link speed topologies. For the 96kbps the scale was increased to 0.125 seconds and the shape remained at 2. By injecting these random latency values into the network, the DCCP data flows experienced congestion similar to those that would be found in real world congested networks. In both variants of the simulation, the same randomly selected latency values were injected into the network at exactly the same time to ensure the comparisons between the standard and modified variant of DCCP were based on identical network conditions.

4.4.3.1 Results of experimentation

Window growth rates in non-congestion networks

This experiment tested whether the standard DCCP model would demonstrate faster send window growth rates when compared to the deadline enabled DCCP model in networks where there was no congestion. In Figures 4.5 to 4.8, the send rate window sizes are shown to compare the different rates of growth experienced by the standard
and modified deadline enabled variants of DCCP.

Figure 4.5: Congestion window growth rates in non-congested networks (Tested at 96kbps).

Figure 4.6: Congestion window growth rates in non-congested networks (Tested at 1024kbps).
After analyzing the results, it was determined that there was no evidence found that suggests the additional overheads needed to incorporate deadline scheduling into the protocol would result in slower send window growth rates in networks where there was no congestion. In all but one of the four simulations (exception being the...
1024kbps topology), the deadline enabled DCCP actually appeared to stabilize before the standard DCCP variant did. The point of stabilization was derived by locating the point where the send window rate varied by less than 3% from the previous reported send rate window size for five consecutive reporting rate periods. Table 4.2 shows the additional results gathered from the experimentation that further support these findings.

<table>
<thead>
<tr>
<th></th>
<th>Std. DCCP 96 kbps</th>
<th>D.E. DCCP 96 kbps</th>
<th>Std. DCCP 1024 kbps</th>
<th>D.E. DCCP 1024 kbps</th>
<th>Std. DCCP 5096 kbps</th>
<th>D.E. DCCP 5096 kbps</th>
<th>Std. DCCP 48776 kbps</th>
<th>D.E. DCCP 48776 kbps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time taken to end slow start</td>
<td>18.8514</td>
<td>17.7507</td>
<td>2.6214</td>
<td>2.705174</td>
<td>1.470166</td>
<td>1.493204</td>
<td>1.020867</td>
<td>1.021309</td>
</tr>
<tr>
<td>Time taken to stabilize in CA</td>
<td>46.6595</td>
<td>44.3667</td>
<td>5.1596</td>
<td>5.4435</td>
<td>2.085311</td>
<td>2.074985</td>
<td>1.25575</td>
<td>1.24046</td>
</tr>
<tr>
<td>Average window size (entire duration)</td>
<td>11237.179</td>
<td>11112.746</td>
<td>116142.317</td>
<td>117669.132</td>
<td>573202.921</td>
<td>588052.464</td>
<td>4893196</td>
<td>5023200</td>
</tr>
<tr>
<td>Average window size (during CA only)</td>
<td>9391.90</td>
<td>9278.34</td>
<td>99797.93</td>
<td>101490.7</td>
<td>533921.9</td>
<td>579455.5</td>
<td>4556197</td>
<td>4800307</td>
</tr>
<tr>
<td>Stdev for avg window size (during CA)</td>
<td>2486.23</td>
<td>2397.79</td>
<td>31063.10</td>
<td>32391.52</td>
<td>116939.8</td>
<td>149087.3</td>
<td>1040223</td>
<td>1152686</td>
</tr>
</tbody>
</table>

Table 4.1: Window growth times in non-congested networks.

From Table 4.1, it can be seen that in three of the four topologies, the standard DCCP model ends the slow start phase before the deadline scheduling DCCP model does. However, the deadline scheduling enabled DCCP model in three of the four topologies is able to stabilize more rapidly than the standard DCCP model tested. The results from Table 4.1 also show that the average window size was larger in three of the tested topologies in both the slow start phase and the entire testing duration in the deadline scheduling enabled DCCP model. These findings show that the additional 16 byte overhead needed by the deadline scheduling mechanism would not have an effect on the growth rate of the DCCP send rate window size in a non-congested network where a constant stream of G.711 packets were traversing the network.

4.4.3.2 Window growth rates in highly congested networks

This experiment tested whether a standard DCCP model would demonstrate faster window growth rates when compared to the deadline enabled DCCP model in highly congested networks. When congestion was added to the network, in three of the four tested networks, there was no point where the send rate was able to satisfy the stabilization requirements used to determine the stabilization point in the non-congested networks experiment (Section 4.4.3.1). With the exception of the 1024kbps
network, none of the other topologies were able to show a point where the send rates for 5 consecutive reporting periods were within 3% of one another. For this reason, a predefined experimentation time was used to end the result collection period in place of the stabilization point. The graphs in Figures 4.9 to 4.12 show the results obtained from the experimentation highlighting the differences in send window growth rates between the standard and deadline enabled DCCP implementations.

Figure 4.9: Window growth rates in highly congested networks. Tested at 96kbps.

Figure 4.10: Window growth rates in highly congested networks. Tested at 1024kbps.
From visual inspection of the graphs (Figures 4.9 to 4.12) and analysis of the corresponding data, it can be seen that in three out of the four tested topologies, the standard variant of DCCP appears to have a larger send rate window size when compared to the deadline scheduling enabled DCCP variant in congested networks. To further support this evidence, Table 4.2 shows the average window sizes for the experiments as well as the standard deviation that occurred during the simulation.
Table 4.2: Window growth times in highly-congested networks.

<table>
<thead>
<tr>
<th>Topology</th>
<th>Std. DCCP 96 kbps</th>
<th>D.E. DCCP 96 kbps</th>
<th>Std. DCCP 1024 kbps</th>
<th>D.E. DCCP 1024 kbps</th>
<th>Std. DCCP 5096 kbps</th>
<th>D.E. DCCP 5096 kbps</th>
<th>Std. DCCP 44716 kbps</th>
<th>D.E. DCCP 44716 kbps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Time taken to end slow start (s)</td>
<td>27.6274</td>
<td>26.9855</td>
<td>3.4530</td>
<td>3.92805</td>
<td>2.639599</td>
<td>2.785893</td>
<td>NA</td>
<td>NA</td>
</tr>
<tr>
<td>Time taken to end of experiment (s)</td>
<td>120</td>
<td>120</td>
<td>60</td>
<td>60</td>
<td>30</td>
<td>30</td>
<td>8.4794</td>
<td>8.4794</td>
</tr>
<tr>
<td>Average window size (entire duration) (bits)</td>
<td>7066.648</td>
<td>6285.322</td>
<td>102634.8</td>
<td>104079.7</td>
<td>197134.1</td>
<td>173463.7</td>
<td>3966568</td>
<td>4643518</td>
</tr>
<tr>
<td>Std dev for avg window size (bits)</td>
<td>2091.074</td>
<td>1883.47</td>
<td>16375.41</td>
<td>17287.14</td>
<td>5564.25</td>
<td>4656.73</td>
<td>1975572</td>
<td>2674908</td>
</tr>
</tbody>
</table>

From the results outlined in Table 4.2, it can be seen that in three of the four tested topologies the average window size for the testing duration was found to be higher in the standard DCCP implementation when compared to the deadline scheduling enabled DCCP implementation. The exception was found in the 1024kbps network where the deadline scheduling enabled topology averaged a marginally higher window size when compared to the standard DCCP model. From these results, it is therefore concluded that there was sufficient evidence found to suggest that the additional overhead associated with the incorporation of deadline scheduling into the DCCP protocol does appear to have a negative impact on CCID3 window growth rates in highly congested networks.

4.4.3.3 Discussion of observed window growth times

Initially it was anticipated that lower average window size growth rates would be a direct precursor to there being longer periods in which it took for the send rate to stabilize and for the protocol to make full use of available bandwidth. In contrast however, as shown in the non-congested network experiment results where stabilization did take place during the simulated time, the average window size does not have any bearing on the time it takes for the rate for the stabilization to take place. In the congested network, the send window rate was not able to stabilize and therefore it is not possible to determine if the time taken for stabilization would have been faster or slower between the models. One fact that was proven during this testing however was that in the congested networks the average window size was smaller for the deadline scheduling enabled DCCP model during the testing period when compared to the standard DCCP model.

Another important thing to note is that the DCCP send rate does not have any
bearing on the actual speed of the network transfer if queuing is occurring on the transmit interface of the sending device. As in this experimentation, the rate at which the DCCP layer passes packets down to the lower layers exceeds the rate at which the network interface can transmit them at. As such the DCCP send rate will have little bearing on the actual transmission rate provided there is a queue of packets awaiting transmission. The only time in which the send window rate would influence actual transmission rates would be where the DCCP layer passed packets down at a slower rate than the transmission interface was capable of sending at. The results shown here indicate that in these scenarios where DCCPs send rate was lower and if there was no congestion on the network, there would be no major or significant disadvantage to the performance of DCCP if deadline scheduling was used. However as soon as congestion appeared on the network, a relatively small reduction in DCCP performance would be expected if deadline scheduling functionality was introduced.

4.4.4 Experiment 2: Throughput, goodput and RTT

This experimentation tests if the overheads needed for deadline scheduling cause a reduction in the throughput, goodput and RTT values during a transfer. As was the case in Chapter 3, throughput is the measure of the average number of bytes or packets being transferred per second by the protocol across the network. The goodput value represents the average number of application payload bytes being transferred across the network per second. The RTT is the time it takes from the moment data is sent to the time acknowledgment for that data is received.

To gather these results a no congestion, no drop network topology was used in the experiments to prevent other influences from affecting the results. The only difference between the two topologies was the use of either the standard DCCP or the deadline scheduling enabled DCCP. As per all other experiments, a range of network link speeds were used to provide a rigorous result set.

In Figures 4.13 to 4.16 the RTT values and the throughput rates (in packets) are presented for each of the tested link speeds. These graphs compare the results gathered from the standard DCCP model to the results gathered from the deadline scheduling enabled DCCP model.

80
Figure 4.13: Throughput and RTT values from experiment comparing standard DCCP to deadline scheduling enabled DCCP. Tested at 96kbps.
Figure 4.14: Throughput and RTT values from experiment comparing standard DCCP to deadline scheduling enabled DCCP. Tested at 1024kbps.
Figure 4.15: Throughput and RTT values from experiment comparing standard DCCP to deadline scheduling enabled DCCP. Tested at 5096kbps.
From the graphs presented in Figures 4.13 to 4.16 it can be seen that in all of the four tested topologies the RTT values are minimally larger for the deadline scheduling enabled DCCP model when compared to the standard DCCP model. In terms of throughput in packets per second, the standard DCCP model was able to produce
a higher rate during the simulated period when compared to the deadline scheduling enabled model. In order to further clarify these results, Table 4.3 shows the summarized results in tabular form gathered from the experimentation. These results include the goodput rate.

<table>
<thead>
<tr>
<th></th>
<th>Std. DCCP 96 Kbps</th>
<th>D.E. DCCP 96 Kbps</th>
<th>Std. DCCP 1024 Kbps</th>
<th>D.E. DCCP 1024 Kbps</th>
<th>Std. DCCP 5096 Kbps</th>
<th>D.E. DCCP 5096 Kbps</th>
<th>Std. DCCP 44736 Kbps</th>
<th>D.E. DCCP 44736 Kbps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average number of packets per sec</td>
<td>50.20</td>
<td>47.76</td>
<td>578.88</td>
<td>537.80</td>
<td>2905.76</td>
<td>2703.48</td>
<td>2389.70</td>
<td>22367.50</td>
</tr>
<tr>
<td>Average throughput (bytes per second)</td>
<td>8654.40</td>
<td>8692.32</td>
<td>99567.36</td>
<td>101106.40</td>
<td>499790.72</td>
<td>508254.24</td>
<td>4110060.40</td>
<td>4167490.00</td>
</tr>
<tr>
<td>Average goodput (bytes per second)</td>
<td>8032.00</td>
<td>7641.60</td>
<td>93620.80</td>
<td>86048.00</td>
<td>464921.60</td>
<td>432556.80</td>
<td>3823312.00</td>
<td>3546880.00</td>
</tr>
<tr>
<td>Average round trip time (RTT)</td>
<td>0.685000</td>
<td>0.712000</td>
<td>0.070000</td>
<td>0.07340</td>
<td>0.02780</td>
<td>0.03610</td>
<td>0.00544</td>
<td>0.00616</td>
</tr>
</tbody>
</table>

Table 4.3: Throughput, goodput and RTT rates.

The results in Table 4.3 show that the deadline scheduling enabled DCCP model actually produced higher average throughput rates (in bytes) for the testing period. This means there is no evidence that the throughput rates in bytes are negatively influenced by the deadline scheduling overheads. This is contrary to the findings in graphs 4.13 to 4.16 that show throughput in terms of packets is reduced by the deadline scheduling overheads.

While the throughput rates may be higher in deadline scheduling enabled DCCP, this value does not represent the amount of application payload data that was able to traverse the network in the given time. As the overheads used to allow deadline scheduling increased the packets size, the increased throughput rate was likely caused by the additional overheads. This does not necessarily mean there was an increase in the amount of application layer data being transmitted. For this reason the goodput rate was used to determine how much application layer data was transferred during the experiment period. From the results it is shown that while deadline scheduling DCCP model produced higher throughput rates, the goodput rates were lower in deadline enabled DCCP than those that were produced by standard DCCP model. In all the experiments the average goodput rates were higher in standard DCCP when compared to the deadline enabled DCCP model (96 kbps = 4.9%; 1024 kbps = 7.1%; 5096 kbps = 6.9%; 44736 kbps = 7.2%). These results show that lower goodput results will be incurred through the use of deadline scheduling.

In addition to lower goodput rates, the average RTT value was also higher in topologies where deadline scheduling enabled DCCP was used. This is not a desirable as it signals longer periods in which it takes packets to arrive at their destination.
and/or to be acknowledged. The reason for the higher RTT value was found to be caused by the fact that the transmission queue size on the senders transmit interface was limited by the maximum number of packets it could hold as a pose to the maximum number of bytes it could hold. As the deadline scheduling enabled DCCP packets were larger, more bytes of data were stored in the queue when compared to standard DCCP when the queue becomes full (deadline enabled DCCP = 100 packets * 188 bytes; Standard DCCP = 100 packets * 172 bytes). Disparity in RTT values only occurred at the point in the experiment when queue sizes began to fill and were most notable where the queue was completely full. Due to the additional serialization time needed to process the larger amount of data, packets entering the deadline scheduling enabled DCCP queue experienced longer wait times before being transmitted. This was the primary cause of the larger RTT values found in the deadline scheduling enabled DCCP model. To test this assertion further, the queue sizes were changed from being limited by a maximum number of packets to being limited by a maximum number of bytes. The results of this experimentation showed that the RTT values were identical for both deadline scheduling enabled DCCP and standard DCCP.

Therefore it is concluded that the results show that the overheads used to enable deadline scheduling decrease the goodput rate offered by the protocol. In regards to throughput, the results show there was an increase in throughput rates found in deadline scheduling enabled DCCP experiments. The increase in throughput however was later determined not to be a desirable characteristic as it was due to an increased rate of overhead rather than an increase in the rate of payload data transfer. In relation to RTT, the standard DCCP protocol provided slightly faster RTT times when compared to the deadline enabled version which would suggest faster packet delivery times could be obtained from standard DCCP when the send device was configured with a maximum send queue size that was measured by packets. There was no detriment found to RTT values when the queue size was specified in bytes.
4.4.5 Experiment 3: Effects of deadline scheduling on packet loss

These experiments investigate whether deadline scheduling reduces DCCP’s ability to react to packet loss. In order to determine this two experiments were carried out. In the first experiment, a single five packet loss event occurs during the simulation and the reaction to this packet loss was measured. In the second experiment a sustained packet loss rate of 5% of the total packets sent was maintained throughout the experiment and the throughput was measured to determine how well the DCCP protocols were able to react to the loss events.

4.4.5.1 Reaction to packet loss events during stabilized CA

In this experiment, the send rate was first allowed to stabilize before the packet loss events were initiated. Once this had been done at a single predetermined interval, five consecutive packets were dropped on the network link between the internet router and the receiver. The time taken for the DCCP send rate to re-stabilize was then measured to determine if the process of sending larger packets would increase the time it took for DCCP to restabilize after the drop event. One of the difficulties in analyzing the results obtained through this methodology was that send rates differed between the standard and deadline scheduling enabled DCCP models throughout the stabilized period. Although the drop events occurred at exactly the same time, in all the experiments, the send rates were different at the point when the drop occurred. For this reason, the time taken for the protocol to re-stabilize after the loss events was used to determine the reaction speed. Figures 4.17 to 4.20 show the send windows measured from the experiments comparing the reaction of the two DCCP models to a single five packet loss event.
Figure 4.17: Graph displaying reaction to 5 consecutive packet loss events in standard DCCP and deadline scheduling enabled DCCP. Tested at 96kbps.

Figure 4.18: Graph displaying reaction to 5 consecutive packet loss events in standard DCCP and deadline scheduling enabled DCCP. Tested at 1024kbps.

Figure 4.19: Graph displaying reaction to 5 consecutive packet loss events in standard DCCP and deadline scheduling enabled DCCP. Tested at 5096kbps.
Figure 4.20: Graph displaying reaction to 5 consecutive packet loss events in standard DCCP and deadline scheduling enabled DCCP. Tested at 44736kbps.

The graphs above in Figures 4.17 to 4.20 show comparisons of the times taken by the standard DCCP model and deadline scheduling enabled DCCP model to stabilize after the drop event. In Figures 4.17 and 4.18, the standard DCCP model is shown to stabilize at a faster rate when compared to the deadline scheduling DCCP model. In contrast however, Figures 4.19 and 4.20 show the deadline scheduling DCCP model stabilizes more rapidly after the drop event when compared to the standard DCCP model. To further clarify the results, Table 4.4 shows in tabular form the times it took the various DCCP models to restabilize after the drop event.

<table>
<thead>
<tr>
<th>Time of packet drop events (seconds)</th>
<th>Std. DCCP 96 kbps</th>
<th>D.E. DCCP 96 kbps</th>
<th>Std. DCCP 1024 kbps</th>
<th>D.E. DCCP 1024 kbps</th>
<th>Std. DCCP 5096 kbps</th>
<th>D.E. DCCP 5096 kbps</th>
<th>Std. DCCP 44736 kbps</th>
<th>D.E. DCCP 44736 kbps</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>50</td>
<td>50</td>
<td>30</td>
<td>30</td>
<td>12</td>
<td>12</td>
<td>7.5</td>
<td>7.5</td>
</tr>
<tr>
<td>Time taken to re-stabilize send rate (seconds)</td>
<td>5.6368</td>
<td>5.7196</td>
<td>0.7896</td>
<td>0.7848</td>
<td>0.2281</td>
<td>0.2249</td>
<td>0.5204</td>
<td>0.5226</td>
</tr>
</tbody>
</table>

Table 4.4: Reaction to packet loss events during stabilized CA.

The results in Table 4.4 show that once CCID3 has stabilized there is no evidence that the additional overheads needed for deadline scheduling will lead to an increased reaction time for the protocol to recover from loss events. In half the experiments carried out the modified deadline scheduling enabled DCCP recovered slightly more rapidly to the lost event occurrences than the standard DCCP model did. It must be noted that the times taken to restabilize are extremely similar (96kbps=0.083 seconds; 1024kbps=0.005 seconds; 5096kbps=0.003 seconds; 44736kbps=0.002 seconds) as shown in Table 4.4. In reality, a multitude of variables were likely to contribute to
these findings. These include, the size of the queue, the point in the acknowledgment window where the loss occurred and the RTT value at the time of the drop. Therefore it is concluded that the results do not show there is any notable difference in reaction time to a single five packet loss event between the standard DCCP model or the deadline enabled DCCP model other than what would occur through random chance based on the timing of the drops.

4.4.5.2 Reaction to packet loss over a sustained period

In an attempt to find a more averaged out and reflective determination as to what effects packet loss had on the two tested models, a sustained packet loss rate of 5% was applied to the network topology. In order to achieve this, the router was configured to randomly drop 5 out of every 100 packets that traversed the network. As the same random seed value was used on both the standard and deadline scheduling enabled models, the same randomly selected packets were dropped on both networks throughout the experiments making the results comparable. The average throughput for the duration of the testing was then measured. In Figures 4.21 to 4.24 the throughput rates recorded during the experimentation are presented to show the difference in throughput rates that occurred during the simulation.
Figure 4.21: Graph displaying reaction to 5% sustained packet loss in standard DCCP and deadline scheduling enabled DCCP. Tested at 96kbps.

Figure 4.22: Graph displaying reaction to 5% sustained packet loss in standard DCCP and deadline scheduling enabled DCCP. Tested at 1024kbps.
Figures 4.21 to 4.24 show the send window rates recorded during the simulation where a sustained packet loss of 5% was configured for the duration of the simulation. From these figures, it can be seen that both models are heavily affected by the sustained packet loss. Because they possess different throughput rates throughout the session, it is not possible to determine if one variant suffers more from the packet
loss than another from these results. However, from visual inspection of the graphs, they both seem equally effected. In order to determine how much of an impact the 5% loss had on the models, the results taken from experiments in 4.4.3.1 were used as a reference. This was done to determine the change that occurred in throughput rates as a result of the loss events rather than trying to compare the results between the two models directly. In Table 4.5 a summary of the results gathered from the experiments as well as the throughput rates from Section 4.4.3.1 is shown.

<table>
<thead>
<tr>
<th>Experiment time</th>
<th>Std. DCCP 96 kbps</th>
<th>D.E. DCCP 96 kbps</th>
<th>Std. DCCP 1024 kbps</th>
<th>D.E. DCCP 1024 kbps</th>
<th>Std. DCCP 5096 kbps</th>
<th>D.E. DCCP 5096 kbps</th>
<th>Std. DCCP 44736 kbps</th>
<th>D.E. DCCP 44736 kbps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Average window size (no drop)</td>
<td>11395.319</td>
<td>11218.457</td>
<td>11335.434</td>
<td>114752.622</td>
<td>565358.350</td>
<td>575914.763</td>
<td>488890.204</td>
<td>4953253.924</td>
</tr>
<tr>
<td>Average window size (5% drop)</td>
<td>9045.819</td>
<td>9266.719</td>
<td>100554.275</td>
<td>103503.630</td>
<td>498882.011</td>
<td>532866.492</td>
<td>482688.695</td>
<td>599530.116</td>
</tr>
<tr>
<td>Difference</td>
<td>2351.500</td>
<td>1951.738</td>
<td>13180.159</td>
<td>11648.592</td>
<td>66476.339</td>
<td>43528.271</td>
<td>4406414.509</td>
<td>4353723.764</td>
</tr>
<tr>
<td>Percentage degradation caused</td>
<td>20.636</td>
<td>17.398</td>
<td>11.609</td>
<td>10.152</td>
<td>11.758</td>
<td>7.558</td>
<td>90.131</td>
<td>87.896</td>
</tr>
</tbody>
</table>

Table 4.5: Reaction to packet loss over a sustained period.

As mentioned previously, instead of directly comparing the throughput rates between the standard and deadline scheduling enabled DCCP models, the results obtained from the experiment in 4.4.3.1 were used instead. From this the change in throughput rates caused by the sustained 5% drop were calculated as a percentage of the previously recorded throughput rate in a non-congested network for the same period. The results show that in all the experiments carried out, the reduction in the average send rate window sizes is less in networks where the deadline scheduling enabled DCCP model was used when compared to the standard DCCP model (96kbps = 3.24%; 1024kbps = 1.46%; 5096kbps = 4.20%; 44736kbps = 2.24%). This shows that the additional overheads used to enable deadline scheduling in DCCP does not cause a reduction in the protocols reaction time to packet loss or to the subsequent re-stabilization duration. This and the previous experiment (4.4.5.1) looking at a single five packet drop event during stabilized flow show that there is no evidence to suggest the additional overheads for deadline scheduling will make DCCP less reactant to packet loss events.
4.4.6 Limitations of experiments

While the multiple iterations of simulations were carried out using a range of varying network speeds in order to reduce the likelihood of such occurrences from skewing the results, it is important to note that the results shown above do have some limitations which may have bearing on the findings above. In the interests of rigor, these limitations will now be discussed.

4.4.6.1 Queue size

One of the limitations of the experiments described in this chapter is that only one queue size was used on the senders transmission interface throughout the experimentation. As shown in Chapter 3, the use of different queue sizes will yield different result in DCCP performance. As all the models in this set of experimentation used a queue size with a maximum of 100 packets the results are consistent across both models. However, these results are not necessarily indicative of what would be experienced if other queue sizes were configured on the senders transmission interface.

The experimentation has shown that in order to reduce the effects the additional overhead has on the DCCP protocol in relation to RTT values, the queue sizes should be limited by a maximum byte size rather than stipulating the maximum number of packets that can be stored in the queue. In doing this, the possibility of larger packets experiencing longer RTT values when compared to smaller packet flows is alleviated. This research has shown that when a queue size is governed in terms of byte size, there is no difference in RTT values between the standard DCCP and the deadline scheduling enabled DCCP models.

4.4.6.2 Determining stabilization points

One of the requirements in the experiments was to determine when the send window rate became stabilized. The reality is such that the send window rate will never become stabilized to a point where it is constant at any point during a transfer. Hence, the point selected as the stabilization point was the first point in the send rate asymptote. The method used was to find where this first point occurred was to

94
determine the first send rate window size where the next five consecutive send rate window sizes varied by less than 3%. In addition, visual inspection of the graphs that were used to showcase the send window rates were also used to narrow down the location where the stabilization asymptote began before using the above test to obtain the exact time.

### 4.4.7 Conclusion of experiments

The results of this experiment have concluded that the additional overheads needed for deadline scheduling do have an impact on the performance of DCCP. The most noticeable of these impacts comes in the form of goodput which was reduced by approximately 7% when deadline scheduling was enabled in the experiment topologies. In addition, an minor increase in RTT was also shown where packet size was used to control the senders queue on the transmission interface. It is suggested that to alleviate this issue, this queue such be governed in terms of number of bytes rather than number of packets. Finally, it was found that the average congestion window growth rate was higher for standard DCCP model than the deadline scheduling model where the network was congested. No such evidence was found in the equivalent non congested network where deadline scheduling enabled DCCP and standard DCCP were found to perform similarly.

In order to be feasible, mechanisms that utilize the deadline scheduling enabled DCCP model described in this chapter should yield at very least comparable performance in terms of goodput when enabled. For example in the topology used above if a mechanism that dropped expired packets was implemented on intermediate devices, the increase in goodput rates or other equivalent performance that resulted from this mechanism would need to be greater than approximately 7%.

### 4.5 Chapter conclusion

The exponential growth in the adoption of real time applications appears set to continue for some time to come. While the current infrastructure supports these appli-
cations, the data they send across modern networks has unique characteristics. The data is time sensitive and possess a finite period in which it is of use to the application. This chapter has introduced a deadline scheduling enabled DCCP protocol where these limited life properties of a real time data packet are incorporated into the transport layer in order to make the transport protocol more efficient at transferring real time data. The incorporation of deadline scheduling functionality into DCCP will eventually lead to the transport layer making more accurate, efficient and rapid scheduling decisions as to how best data should be transferred across the network to ensure the greatest good to all applications sharing the infrastructure. The additional dimension of information will allow existing theoretical and new scheduling techniques to be employed on congested intermediate devices. For example, the scheme will allow the intermediate devices the ability to purge stale data packets that are no longer useful to the application. In doing this queue sizes can be reduced to ensure other packets are given the best chance of arriving within their respective lifetimes. This chapter has shown not only the benefit that would be added by the proposed scheme, it has also demonstrated how the deadline scheduling is incorporated into DCCP in terms of both the actual DCCP standard as well as providing an actual working simulation model of the deadline scheduling DCCP protocol.

While the scheme does create the possibility for many new and exciting methods of making real time data transfers using DCCP more efficient, the additional overheads needed to implement the scheme do undeniably cause some detriment to the performance of DCCP. The amount of detriment caused by the additional 16 bytes of overhead was measured through a series of experiments. This detriment should be considered as worst case scenario. If implemented as an IETF standard, the scheme would likely utilize a smaller variant of the NTP timestamp.

The following chapters will show how the option fields will be utilized to provide more efficient transfers of real time data. In Chapter 5, five packet purging schemes will be described that delete stale packets on intermediate devices in an effort to free network resources for non-stale packets. In Chapter 6, a novel probability based equation that uses information from the routing protocol to determine whether a packet will arrive within its useful lifetime will be showcased. Both the mechanisms in Chapters 5 and 6 require the information stored in the additional option fields made available by the deadline scheduling DCCP model described in this chapter.
Chapter 5

Lifetime Discard Protocols for DCCP

5.1 Introduction

This chapter describes five lifetime packet discard mechanisms for the deadline scheduling enabled DCCP protocol. The purpose of each of these lifetime discard mechanisms is to detect and discard real time packets that have exceeded their useful lifespan instead of allowing them to continue to traverse the network. This chapter is divided into two parts. In the first part the theory relating to the lifetime packet discard mechanisms described in this research is presented. This includes the background, motivation and need for such schemes. Following this, the five unique packet lifetime discard schemes created specifically for the deadline scheduling enabled DCCP protocol described in Chapter 4 are presented. In addition to how each of these schemes work, their respective advantages and disadvantages are also defined from a theoretical point of view.

In the second part of this chapter, the stability, effectiveness and benefits of the five purging mechanisms are then quantified and compared through experimentation to determine which mechanism yields the most benefit to the deadline scheduling enabled DCCP protocol. In addition to reduced bandwidth consumption through alleviation of expired packets, this chapter also shows how more bandwidth is made available to non stale packets through the various mechanisms. Finally the chapter
explores the critical issue of fairness that emerges between competing DCCP flows when the discard mechanisms are employed.

5.2 Background

In the previous chapters (Chapters 1-4) the notion of real time packets possessing finite lifespans was discussed extensively. In addition to this, the potential for being able to use this information to make the transportation of these packets more efficient was also alluded to. Although in Chapter 4 it was shown that incorporating deadline scheduling into DCCP added slightly to the overheads associated with the protocol, having the application layer specify a maximum TTL value and embedding packet’s birth time into the DCCP header allows intermediate devices the opportunity to detect packets that have exceeded their useful lifespan. In this chapter packet life discarding schemes that take advantage of this newly founded ability are demonstrated.

These packet discarding schemes will deal purely with packets that have exceeded their advertised useful life period as defined by the application sending those packets. Once the packets have reached this age, they are no longer useful and would likely be dropped by the application layer upon arrival. For the purposes of the research described in this chapter these packets will be referred to as stale packets. Instead of allowing stale packets to continue to traverse the network, intermediate devices take various actions on stale packets such as deleting or marking the packets. By doing this, the mechanisms ensure precedence and bandwidth resources are given to fresher/useful packets when congestion is experienced and that packets that are no longer useful do not contribute to existing congestion.

5.3 Scope of the problem

Before commencing discussion on how the packet life discard mechanisms operate, this section first describes the causes of stale packets in a network. Essentially stale packets can occur due to one of two reasons. The first reason is that the application sets an unrealistic maximum TTL value in the packet options field. Unrealistic here
refers to setting a maximum TTL value that is less than the cumulative latency found in the network during normal or optimal performance periods. An example of this would be if the network possessed a minimum latency of 150ms due to packet serialization and transmission (excluding any contention) and the application set a maximum TTL value of 100ms. The result would be that all packets received by BOD devices beyond the 100ms latency boundary would be deemed as stale. This phenomenon can be avoided by correct specification of the maximum TTL value by the application.

The second cause of stale packets is through the occurrence of congestion events in the network where queuing of packets takes place and where packets exceed their useful lifespan as a result of the delays associated with the queuing processes. As shown in Chapter 3, when the rate of packets entering an intermediate device exceeds the device’s ability to forward those packets on, queuing of packets is necessary. As queues become larger, so too does the time it takes the packets to be delivered to their destinations as they are subjected to the delay associated with having to transmit all the preceding packets in the queue.

DCCP CCID3’s congestion algorithm is designed in such a way that it will continue to increase the congestion send rate window up until the point that it oversaturates the network and receives information in the form of acknowledgment of lost packets or packets marked with ECN [49] nounces. Once it receives either of these packet types, the rate will then begin to stabilize. The process of probing the network for available bandwidth in order to determine the correct send rate window causes congestion on the network, not only for the flow directly but for all other flows utilizing the shared infrastructure. It was also shown in Chapter 3 (Section 2.7) that even once the send rate for a flow had stabilized, the CCID3’s congestion algorithms so called “stabilized” send rate continued to causes stress on the network as evidenced by the continual increase in the RTT values over the duration of the flow after the initial congestion avoidance spike. In Figure 5.1 (extracted from experimentation conducted in Chapter 3), the congestion avoidance congestion spike is clearly reflected in the RTT value between 5 and approximately 22 seconds. Following this, it can be seen that even once the equation based algorithm is activated the RTT value continues to grow indicating a continued increase in congestion. Eventually this increase in the RTT value would cause the congestion algorithm to reduce the send window size
which would decrease the RTT value by lowering congestion.

Regardless of the cause of the congestion, if the combination of the minimum latency of the network combined with the added delay caused due to congestion exceeds the packet’s maximum TTL value, then the packet will become stale. As stale packets appear inevitable, mechanisms are needed to reduce the number of packets becoming stale in order to ensure data received remains meaningful to the applications. If the congestion algorithm maintains congestion levels that are acceptable to the application, then the packet life discard mechanisms described in this chapter should not be triggered. If however the packets traversing the network are found to be stale then action will take place on the packets. This is done to ensure the congestion window is reduced by the sender to assure timely delivery of subsequent packets by reducing the size of the queues causing the packets to become stale.

5.4 Overview of the packet life discard mechanisms

The functionality added by the modifications to the DCCP protocol described in Chapter 4 is used in all of the packet life discard mechanisms described in this chapter by intermediate devices (BODs) located between the sender and receiver in order to provide more efficient and timely transmission of real time data. When a packet arrives at a BOD, the birthtime and maximum TTL variables are extracted from the DCCP packet’s options fields (described in Chapter 4 Section 4.2.2.4). The birthtime and maximum TTL values are then added together to determine the packets expiry time. This value specifies the time at which the packet’s contents will no longer be useful to the receiving application. The DCCP process compares the current system
time (NTP synchronized) to the packet’s expiry time to determine if the packet has become stale during transit. If the packet is found to be stale one of five mechanisms for dealing with this stale packet is then used. If the packet is not found to be stale, then it is allowed to pass through BOD without manipulation/ modification. In the following section, the five packet life discard mechanisms created specifically for the deadline scheduling DCCP protocol will be described.

5.4.1 Deletion method for purging stale packets

The first DCCP discard mechanism method known herein as the “basic deletion mechanism” simply deletes any packet in a given queue if it is found to be stale. By deleting stale packets, more bandwidth and shorter queues are afforded to other non stale packets as an immediate consequence of the deletion action. As a secondary consequence to the flow where the stale packet was been deleted, the deletion action will appear to the receiving device as if the packet has been lost and therefore no acknowledgment will be sent back to the sender for that packet. As a result the DCCP congestion algorithm on the sender will take effect and the send rate will be reduced. This will slow down the number of packets the sender is sending into the network and reduce the congestion levels. Inspiration for this scheme was derived from worked carried out in [26, 15].

While all packet loss events will trigger a reduction in the DCCP send rate, this reduction is dependent on how long the flow has been established. Results from Chapter 3 (Section 3.6) indicated that when the equation based flow control algorithm has taken effect and the flow has been established for a long duration, the effects of the deletion or loss events will be less significant when compared to a flow that has not been established for a long period. This is because the younger flow’s congestion control equation has a limited number of variables to determine the appropriate send rate and is normally still trying to stabilize in congestion avoidance mode. The pseudocode in Figure 5.2 shows a simplified version of how the deletion method is implemented on BODs.
As shown in Figure 5.2 when a packet is received it is checked to see if the expiry time value has elapsed. If it has, then the packet is deemed to be stale and is deleted. If the expiry time has not yet passed then the packet is forwarded to the outgoing transmission queue for delivery.

5.4.1.1 Advantages of the scheme

The key advantage of this discard mechanism is that the processes are computationally simple and can be done when the packet arrives, during queuing, just prior to transmission of the packet and/or any combination of these. Deleting stale packets saves the network from transmitting redundant information and will result in the same reduction in send rate window reduction as the more elaborate mechanisms which will be discussed later. In addition this discard mechanism does not manipulate any of the data or header fields and therefore would allow current and future real time encryption schemes to work in conjunction with it as no packet content is changed. Finally unlike the other four packet life discard mechanisms, this mechanism does not rely on the ECN protocol.

5.4.1.2 Disadvantages of the scheme

One of the problems with this scheme is that flows that experience stale packets entirely or mainly due to network latency and partially because of congestion could potentially never be able to establish a session if this scheme was used. For example
if the application sets the maximum TTL value on the packets for a flow to 150ms and the default latency on the network is 160ms, then all the packets for that flow would be deleted. If the applications are made responsible for the setting of the maximum TTL value then this may not necessarily be a disadvantage. Inability to create a session may signal to the application that it should lower its expectations by changing the codec or increasing the maximum TTL value on the packet.

Another disadvantage is that the deletion method mechanism could also potentially cause a premature end or shortened duration to the congestion avoidance phase of a flow’s initial window growth period which may result in a longer time being needed for the algorithm to utilize maximum available bandwidth across a network. This is because the probing activities which the CCID3 algorithm employs at the beginning of a session to determine the network conditions may be abruptly ended as a result of a purge event resulting in the limits of the network never being fully tested.

Another disadvantage with the basic deletion method is that it can only take action on the particular flow on which the stale packets are found. The action of deleting a packet and reducing the send rate will not have any effect on the other streams other than to reduce overall congestion levels. In some cases these other streams may even be responsible for this congestion and in some cases the wrong flow could potentially be punished incorrectly for congestion it does not create. This basic deletion method may also lead to a scenario where DCCP streams are not as competitive as other transport layer streams such as UDP or TCP which do not have discard actions taken on them. This is because when congestion is experienced the send rates of these non-controlled protocols will not be slowed as quickly as the DCCP flows if they are even slowed at all.

Probably the most major of the drawbacks found in the basic deletion mechanism is that there is potential for a large number of simultaneous packets to be dropped when the combination of the delay resulting from increased network congestion, and the minimum network latency, causes multiple packets in a flow to become stale. Dropping a large number of simultaneous data packets for a prolonged period causes instability in the DCCP protocol and can lead to the session timing out. In some instances where packets are dropped and not received, the various timeout mechanisms and in particular the no-feedback timer in DCCP will need to expire before action is taken
to reduce the send rate. This period potentially allows an unacceptably large number of packets to be dropped while the timer counts down. The configuration of the DCCP history discounting mechanism set in place to combat sudden burst of losses and limit bursts of congestion from rendering the send rate to virtually nothing may not sufficiently react to the extent of simultaneous losses that may occur as a result of the basic deletion mechanism. Jacobson’s [48] work could be used to limit this in future work.

Any discarding mechanism should be able to allow enough traffic through the network in order to allow the packet loss events to be detected rather than relying on no-feedback mechanisms in order to avoid out of bound synchronicity errors from occurring. For this reason slightly more sophisticated deletion mechanisms will now be described which aim at overcoming these issues.

5.4.1.3 Additional variations explored on the basic delete mechanism

From the trial experimentation carried out in the research and through implementing the various discard models into BOD devices on the simulated experiment network topology, it quickly became apparent that two relatively simple changes to the basic deletion mechanism could be made to reduce the likelihood of a timeout event from occurring when a sustained number of deletions occurred.

Having found a topology with characteristics that would cause enough drop events to result in a flow to timeout, changes were then made to the deletion mechanism to see if they would prevent the DCCP session from collapsing. In the first of these changes it was decided a more moderate deletion approach should be used where instead of deleting every stale packet that was detected only a certain ratio or percentage of stale packets that were detected were discarded. The results obtained in the trial proof of concept topology showed that configuring the basic deletion mechanism to discard only three out of every five packets stale packets detected and by leaving the other two packets unchanged (i.e. sent stale) was adequate enough to prevent the session from timing out in conditions where it had previously done so. It is clear the ratio of stale packets that are not deleted by the protocol would need to be mathematically calculated in order to ensure the receiver was able to send adequate acknowledgments to the receiver in time to prevent the timeout events from being
triggered and at the same time offer optimal deletion rates.

Having determined that timeouts could be avoided, a second and even more advanced solution to resolving the timeout problem was also trialled through simulation and found to be effective. Instead of dropping packets based on a ratio as described, this mechanism utilized a timer and counter variable to ensure that only a fixed number of stale packets were discarded per round trip period. In a similar mechanism to DCCP's own history discounting scheme [8], this solution prevented situations where sudden fluctuations in congestion levels caused a series of packets to become stale which would have been deleted by the discard mechanism, causing the flow to collapse. The counter in the test topology was set to delete a maximum of three stale packets per round trip time period. This would mean that the first three stale packets detected in every round trip time were discarded and then the rest of the stale packets in that same round trip time were allowed through onto the transmission interface unchanged.

From the results obtained from the trial experiments, both of these mechanisms were not nearly as efficient in removing stale packets from the network as the original basic deletion method as they allowed certain numbers of stale packets to traverse the network. However they did ensure the DCCP session did not time out when congestion occurred in the network. As mentioned these trials were carried out as proof of concept experiments and the selection of delete ratio and maximum number of packets that were deleted per round trip time were not optimal. As will soon be shown, a more advanced mechanism which makes use of the basic deletion method in an alternative way was developed and this meant that further research into determining these optimal values was no longer warranted.

With the reduction of the efficiency these changes caused to the basic deletion mechanism it was decided that the simulation topology used to compare the various discard mechanisms in later experiments described in this chapter was to be constructed in such a way that timeouts did not occur. This was done by selecting specific link speeds and queue sizes to ensure the lost packet events were reported back to the sender before catastrophic events that would cause the session to timeout could occur. For this reason, the most efficient form of the deletion discard mechanisms was used in the comparison experiments in Section 5.8 (i.e. where every stale packet detected was deleted).
5.4.2 ECN used as a stale packet control mechanism

While the ECN protocol was not specifically designed for the purpose of alleviating stale packets in a network, its original purpose of reducing congestion as well as the fact that it is built into the DCCP standard means it can easily be adapted for this purpose. In this mechanism stale packets are marked by ECN in order to trigger a reduction in the sender's send rate thereby reducing the congestion on the network that is causing the packets to become stale. This reduction occurs because DCCP is configured to react to ECN marked packets in the same way it would had the packet been dropped or lost. Specifically this mechanism causes the sender to slow down its send rate upon receiving an acknowledgment with congestion experienced (CE) event reported. Using ECN in this manner does not remove the stale packets from the network and therefore is not as efficient at instantly reducing congested queues when compared to the basic deletion method. However the advantage with reducing the congestion rate in this manner is that it keeps the DCCP acknowledgment information intact and prevents timeouts or session collapses like those found in the basic deletion mechanism. This creates a tradeoff scenario where ECN maintains reliable transport layer properties but subsequently is not as efficient at reducing the need to still transfer the redundant/stale data packets across a congested network. By not deleting the stale packets from an intermediate queue, the overall ability for the network to transfer packets for all streams may be hindered. Figure 5.3 presents the pseudocode used to implement the ECN packet control mechanism.

![Figure 5.3: Simplified pseudocode for pure ECN stale packet control mechanism.](image)

As can be seen in Figure 5.3, when the BOD detects a stale packet, it marks the packet with an ECN flag of 11 in the ECN option field at the IP layer (as per
normal ECN operation). The ECN flag of 11 is then forwarded back to the sender in the acknowledgment sent by the receiver signaling congestion was experienced. This event according to the ECN standard must result in the same congestion control response as a single dropped packet would result in [49]. However, when it comes to how the receiver should react to receiving a packet marked with a CE flag, there is some ambiguity between the CCID3 standard and the DCCP standard as the DCCP RFC [4] states “Receivers SHOULD send acknowledgments immediately on receiving packets marked ECN Congestion experienced. However, there is no need to send such immediate acknowledgments for marked packets more than once per round-trip time.” However in the CCID3 RFC [8], there is no mention of triggering an acknowledgment upon receiving a CE marked packet. Instead the CE marked acknowledgment packet is only triggered once per RTT when an ECN flag of 11 was detected. For the purposes of this research existing Linux kernel implementations of DCCP were examined and were found not to trigger any instantaneous acknowledgments when the first ECN marked packets were received. For this reason the simulated DCCP model was also configured to not trigger any instant acknowledgment when ECN packets flagged with a value of 11 were received. As per the CCID3 standard, the sender was configured to reduce the send rate in the same manner as it would for a lost or dropped packet.

One advantage of this mechanism is that it does not hinder in any way normal ECN functionality. If an alternative active queue management scheme is located on an intermediate device further downstream from the BOD and is experiencing unsustainable queue sizes, then the stale packets marked as CE by this scheme are perfect candidates to be dropped.

Another advantage with this scheme is that if more intervention was required or instant feedback needed based on deleted packets, then a simple add-on to the ECN module placed on the receiver could easily be appended to the current DCCP standard to monitor the number of CE marked packets and act in any manner desired. For example an ECN-application layer specific add-on could be appended to the ECN module that allowed ECN to communicate to the application that congestion was occurring and that a reduction in the senders send rate was imminent. The application could then adapt its jitter buffer or move to a different codec temporarily. While these options were not explored, this scheme does offer a number of such possibilities which are quite attractive.
One of the major limiting factors in using the ECN protocol in the past has been that its adoption rate has been relatively slow. If DCCP is being used then this problem no longer exists as ECN is built into DCCP and therefore the use of DCCP implies by default ECN capability. Other criticisms in relation to ECN use have related to misbehaving nodes not responding to ECN notifications. Through the utilization of ECN nounces and the resultant penalties for devices found to be misbehaving most of these issues have now been resolved. In this mechanism the ECN protocol was not altered at all therefore the anti-misbehaving node detection via ECN nounces would operate as defined RFC 3168 [49].

5.4.3 Phantom packet method for purging stale packets

A third proposed method known as “phantom packet purging” will now be presented. The basis for this was originally conceived by Gurtov and Ludwig in [26]. Note Gurtov and Ludwig refer to this mechanism as header-casting however for the purposes of this chapter the technique will be referred to as the phantom packet purging method. Phantom packet purging is designed to act in a more sophisticated manner to the pure ECN and basic deletion methods. The theory is that instead of simply marking stale packets or dropping them entirely, the data portion of the packet is stripped away and the header portion of the packet is allowed to continue through the network and on to the receiving device’s application layer. This reduces the size of the packet and prevents redundant data from traversing the network. At the same time this mechanism allows the DCCP congestion control signaling mechanism to remain intact thus ensuring more accurate reporting information is available to the protocol in relation to network conditions and also prevents timeouts from occurring. In Figure 5.4, the pseudocode for the phantom packet model is presented.
Figure 5.4: Simplified pseudocode for phantom packet purging mechanism.

Upon receiving a stripped down stale packet, the receiver could be configured to react in a number of ways. Due to DCCP’s strong association with the ECN protocol it seems apparent ECN should be used by the receiver to signal back to the sender that congestion events have occurred and that the send rate should be reduced.

It is also noteworthy that during the design of the phantom mechanism the Drop code option in the DCCP RFC 4342 [8] was considered as a method of marking packets which had been stripped of data. According to the standard, when the Drop code is set to the value of 2 (receive buffer drop), the send rate is reduced by one packet per RTT for every packet that contains a Drop Code 2 option field. Further investigation into using this option field however revealed it was not ideally suited to the role due to its original purpose. The option 2 field is designed in the DCCP protocol to be used by the receiver when a packet loss occurs due to the receiver buffer overflowing. For each overflow event that occurs a new acknowledgment for that packet is generated by the receiver and sent to the sender containing the Drop option field set to 2. Given this purpose, the standard makes no mention as to what actions a receiver should take if it receives a packet with a Drop option field set to 2. Testing in the simulation model revealed that unless the DCCP protocol was modified to explicitly check this option on incoming packets, the default behavior of a DCCP receiver is not to check for this field as it would not expect to receive Drop 2 options. As such if no buffer overflow occurs then the action of marking the packet at an intermediate device is essentially pointless. In order to make use of this option, the default DCCP standard would need to be changed which is against the principles that govern this research’s objectives, that is, of creating additional modules that can
work in conjunction with the existing DCCP standard.

The biggest advantage to utilizing a phantom packet mechanism is that it keeps the end-to-end semantics of the DCCP layer intact and theoretically should prevent timeouts from occurring. It also seems intuitive that if packets have been dropped because they have become stale then this should be reported to the receiver. While it seems like a fairly trivial process to strip out the data portion of the packet and forward on only the header portion to the receiver, the reality is such that this process is markedly more complex to implement than the previous two mechanisms. The first problem with stripping the data portion out of the packet and leaving the header portion intact is that the CRC checks at each of the layers need to be recalculated before the packet can be transmitted. Failure to do this may result in the modified packet being dropped for having changed in size when CRC checking occurs at intermediate or the destination devices. While it is possible to recalculate the CRC, it does bring into question computational requirements and the ultimate scalability of the scheme.

Probably the most important disadvantage to the mechanism is that it is not currently possible to implement the mechanism in the current or deadline scheduling DCCP standard because of the methodology DCCP uses to determine packet loss events. In an effort to explain this assertion further and sufficiently, the attempts to implement the phantom discard scheme into the experiment's simulation model will now be presented to show the reasons why the scheme is not possible in DCCP.

In the first instance of the attempted implementation, the model was configured to receive a stale packet, strip out the data portion and then pass unmodified header onto the outgoing interface. As an initial shortcut to first see if the scheme was feasible, CRC checks were disabled on all layers of the devices and no recalculation of the CRC was performed. The act of stripping out the data portion and still reporting the original unmodified sequence numbers in the DCCP header was found to cause major instability in the DCCP protocol if the data portions that were being reported as being present had indeed been stripped. In the simulated model used in this research, memory violations in the simulator were caused when the phantom packet was received as there was no way for the DCCP protocol to identify that the data portion was missing as the header appeared normal. When the DCCP model tried to access the data portion of the DCCP packet, the simulation crashed.

Although this was a simulation error, it is reasonable to assume a real world
implementation of the DCCP protocol would also encounter serious instability if this occurred. In addition, the DCCP protocol relies on header information solely to calculate loss events. With an unmodified header, the DCCP protocol assumed that the missing portion of the packets were in fact present and therefore no loss events were added to the loss event table. Essentially this led to a situation where even though the data portion of the packet was not present, the phantom packet was causing DCCP to act in such a way that the data was believed to have been received successfully and therefore no loss events were reported and no reduction in the send window resulted (via loss interval acknowledgment number). This was completely opposite to the desired goal of the mechanism.

From this it became clear that the phantom discard packet mechanism should be used instead to inform the DCCP receiver of the fact that a stale packet’s data had been removed. This was to be done via the phantom packet. After several changes to the protocol, a solution of creating phantom packets was derived which involved modifying the original packet so that it would report the stripped data events to DCCP protocol. This packet was generated using the header portions of the stale packet so the receiver could identify the correct session flow to which the flow belonged, followed by information pertaining to the data that was dropped.

Having created this new packet type which contained information relating to the stripping event and in particular the sequence numbers of the stripped packets, focus then shifted to modifying the DCCP protocol to firstly detect such packets upon arrival and secondly take action based on the information stored in them. This was the point at which it became clear that such a phantom discard mechanism was not possible in DCCP.

When a packet was detected with a newly created option field created to facilitate detection of phantom packets, (option was received with a value of 1), the data portion of the packet was to be accessed and the sequence numbers for the lost data extrapolated. Following this, these losses were then to be applied to the loss table. When trying to make use of these sequence numbers, it became evident that they could be of no use to the DCCP protocol due to the way in which CCID3 detects lost packets events. When determining whether a loss event has occurred in order to calculate the loss value (p), the CCID3 protocol only considers a packet loss to have occurred when three packets with a higher sequence number are received. This
the method of determining packet loss is so intrinsically involved in the process of determining the various parts of loss interval lengths that are used to determine the loss rate, that to change the DCCP protocol so that it would be able to be informed of lost packets rather than detecting them in the way described above, would require an entire re-write of all the loss interval length calculations. For this reason, the phantom packet mechanism described herein was deemed infeasible and will not be included in the comparison of discard mechanisms. This is because to implement this scheme, major fundamental changes to the DCCP protocol would be required. This violates one of the key goals of this research which was to build discard techniques that did not require changes to the current DCCP and CCID3 standards.

5.5 Hybrid schemes

As explained in the preceding sections, the above three methods of discarding packets have previously been explored, albeit using mostly theoretical approaches and not specifically for the DCCP protocol. None the less they have shown they have potential for reducing congestion and the effects of stale packets. The fourth and fifth mechanisms that will now be described are novel packet discarding mechanisms for DCCP. Both these mechanisms will make combined use of both the basic deletion mechanism and the ECN protocol in such a way that they alleviate the shortcomings experienced when used individually to reduce the effects of stale packets on a congested networks. The first of these two models will be referred to as the “advanced deletion method”. The second of these models is called the “preventative hybrid method”.

5.5.1 Advanced deletion method for purging stale packets

It was theorized that by incorporating the ECN protocol into the basic deletion model described above, the same reduction in send rate for that flow could be obtained on the device that would occur by simply dropping the packets and at the same time this would eliminate the potential for timeouts to occur. Instead of dropping the packets, or leaving stale packets on the network to avoid timeouts as was the case before, a ratio
of packets were instead marked with a ECN CE flag and the remainder of the stale packets were dropped. In doing this the potential timeouts experienced in the basic deletion mechanism were avoided as enough packets were received to keep the DCCP congestion control system intact. By dropping the other packets, there was also an immediate reduction in the need to transmit as many stale packets. In this scheme three out of every five stale packets were marked with the ECN flag set at CE and the other two stale packets are deleted. No stale packets were allowed to traverse the network unchanged. The ratio of packets marked by ECN or dropped is configurable in this scheme which allows network engineers the ability to tailor the ratio to a particular network to obtain a particular result. The pseudocode representing this scheme is shown in Figure 5.5.

```
Get birthtime and MLT from DCCP options field;
Expiry time = birthtime - MLT;
Ratio_Count;

If Expiry time > current time then && ratio_count < 3
  Mark the stale packet with CE ECN flag (OPTIONAL);
  Pass modified packet into the outgoing queue for delivery;
Else if Expiry time > current time then && ratio_count >= 3
  Delete the packet;
  Ratio_Count ++
  //Reset ratio count after 2 iterations
  If Ratio_count = 5
    Ratio count = 0;
End if
```

Figure 5.5: Simplified pseudocode for advanced deletion mechanism.

In addition to removing the timeout problem and removing some of the stale packets from the network, this scheme also has the advantage that if an end node or device misbehaves and ignores the ECN messages, the deletion event will still force the send rate to be reduced. This adds to the protection ECN has against misbehaving ECN nodes through the use of ECN nounces. The disadvantage with this mechanism is that a high number of packets carrying expired data are allowed to pass through the network, when compared to the basic deletion mechanism, even though the packets will be dropped at the application. The next mechanism discussed solves this problem.
5.5.2 Preventative hybrid method for purging stale packets

As the name suggests, this mechanism has a preventative element that tries to prevent stale packets from occurring by monitoring the age of packets throughout the flow. To do this, when the age of the packets in a flow reaches a certain predefined threshold (i.e., 90% of the expiry time), ECN is used to mark these packets which have ages greater than this threshold as having experienced congestion. This is done by setting the ECN flag in the packet header to CE (11). Once received, the receiver activates the CE flag in acknowledgments which are sent back to the sender. Upon receiving these acknowledgments and subsequent ECN CE flags, the sender reacts as if a packet loss has occurred and reduces the send window accordingly as per normal ECN behavior. In doing so, senders can be warned about impending stale packet events caused by increased congestion before they occur and lower their send rates accordingly to reduce congestion and RTT values. This is done in an effort to prevent stale packets from ever occurring on the network in the first place. The inspiration for this threshold notion was derived from the RED mechanism in [21]. In the hybrid deletion mechanism the ECN marked packets still carry useful data. Unlike phantom packets, the entire packet is allowed to traverse the network as it still contains data that has not expired and therefore is useful. If the ECN fails to prevent stale packets completely, then deletion occurs only on packets that are found to be stale (older than expiry time) which are fewer in volume based on the preventative actions taken before conditions in the network become such that stale packets occur.

The pseudocode for the preventative hybrid mechanism is presented in Figure 5.6.
As previously mentioned while prevention is the aim of this mechanism, if ECN is unable to prevent stale packets entirely, then when the hybrid mechanism detects a packet with an age above the maximum allowed age the packet is simply dropped as per the basic deletion mechanism. The process of generating phantom or ECN marked packets for stale packets creates unnecessary overhead as the action of simply dropping the packet will yield exactly the same result and at the same time remove the need to further transmit redundant data across a congested network.

One of ECN’s major weaknesses is that misbehaving end points can ignore CE flags and not reduce their corresponding send rate when they receive a CE marked acknowledgment. With this hybrid model, misbehaving nodes can still exist, however if the node continues to expand its congestion window and packets begin to experience excessive delays in the network due to congestion caused by the sender not slowing down or being warned to slow down then the packets age may grow beyond its useful
lifespan while being transported across the network. By dropping these stale packets, the incentive for non-behaving nodes to exist is removed as the effects of multiple packet drops may be far more extreme than sensible ECN reduction of congestion window size.

The difficulty in implementing this mechanism is that by setting the threshold when ECN marking starts to take place to a value that is too low, there is a possibility of creating an artificial throughput cap as a result of triggering reductions in the send rate. If the send rate is lowered before the links are fully utilized because of ECN marking taking place too early, then this would result in inefficient utilization of the network. At the same time setting the threshold too high may result in there not being enough warning time to allow the ECN process to inform the sender that it should slow its send rate. Setting the threshold too high could lead to higher levels of stale packets occurring. In order to gauge roughly whether a more aggressive ECN threshold or a less aggressive threshold provides better performance, two variations of the preventative hybrid mechanism are tested in the experiments carried out in Section 5.7. The more aggressive threshold is set at 80% of the packet's reported maximum age. The less aggressive threshold is set at 90% of this same value. This threshold determines the point at which packets begin to be marked with ECN CE flags. Again these threshold values can be configured by network engineers on a per network basis as required. Instead of a utilising a threshold, the expiry time could simply be set to 90% of the the calculated value to reduce computation. Utilising thresholds are however more in line with other congestion algorithms and hence the reason to utilise them was made.

5.6 Acceptance of a discard scheme

The clear danger with any of the packet discard mechanisms described in this chapter is that applications (more appropriately application programmers) may feel that the purging steps are draconian and as a result may then begin setting their maximum TTL values to extremely high values in order to ensure their packets do not expire. While there is nothing stopping them from doing this, all this action will yield however is a network with conditions no worse than what currently exists, that is, networks

116
where a packet’s freshness information is not scrutinized.

In addition, there is a strong need for any of these mechanisms to operate correctly in the presence of other protocols. If the event arises where either DCCP streams or TCP and UDP streams are allocated unfair access to shared network resources, then it is highly likely the scheme will not be adopted readily. Fairness of the mechanism was tested in the experimentation described in Section 5.7 to determine if the discard mechanisms are feasible and provide fair allocation of bandwidth. It is important to remember at this point that the goal of these mechanisms was to provide more resources to flows carrying useful non-redundant information during times of high congestion. When there is minimal congestion on a network, it is recommended that the discard mechanism be turned off.

5.7 Comparison of the effectiveness of the discard mechanisms

5.7.1 Experimentation aims

The experimentation described in this chapter has two main objectives. Firstly, the experimentation investigates if the proposed packet discarding mechanisms described in Section 5.4 provide better performance than the current approach where no action is taken (i.e. default DCCP). The key indicators that are used to measure this improvement include overall throughput rates, goodput rates and the effects of the discarding mechanisms actions on the DCCP congestion window. Secondly, the experimentation explores fairness amongst multiple simultaneous flows when a discard mechanism is enabled. This is done to determine if the actions of deleting stale packets or marking them using ECN have an effect on a flow’s ability to obtain a fair proportion of available network resources.
5.7.2 Experimentation design

5.7.2.1 Problem scope definition

It is important to remember throughout the following sections, that the results obtained from the experimentation are heavily dependent on the network topology and the simulated network conditions. The simulated topology used to collect results for these experiments was designed to provide a worst case scenario topology in order to highlight the effectiveness of each mechanism. The experiments therefore focus primarily on obtaining a collection of proof of concept results in severely congested networks, where packet discard mechanisms are most effective.

5.7.2.2 Experiment scenarios

For the proof of concept experimentation carried out to determine the effectiveness of the discarding/purging mechanisms described in this chapter, two groups of simulated topologies were created. The first topology group type was used for single DCCP flow analysis. As such, it contained only a single DCCP sender and a single corresponding DCCP receiver. The second topology type was designed to explore the performance characteristics of the discard mechanisms when multiple flows existed on the network. In the multiple flow topologies, two and three simultaneous flows were configured to pass their packets through the same shared intermediate infrastructure. To create the two or three required flows, either two or three individual sender and receiver pairs were used respectively. The network infrastructure located between the senders and the receivers remained the same for both the single and multiple flow topology groups. Figure 5.7 shows the two topology group types graphically.
Once the topology groups had been created, six variations of each of the topologies were then created. In each of these variations, the only characteristic or simulation property that was changed on the topology was the stale packet discarding mechanism used. As no other characteristics or system variables were changed between the topology variations, any changes found in the performance could therefore be attributed directly to the discard mechanism selected for that particular variation.

In the first variation of the simulated topology, no packet discard mechanism was used in order to gather a baseline performance model of standard DCCP. In the second variation of the simulated topology, the basic delete packet purging mechanism (Section 5.4.1) was used exclusively. In the third variation of the simulated topology, the pure ECN packet discard mechanism (Section 5.4.2) was used. In the fourth vari-
ation of the simulated topology, the advanced delete mechanism (Section 5.5.1) was employed. In the fifth and sixth variations of the simulated topology, the preventative hybrid mechanism (Section 5.5.2) was used. In the fifth variation an ECN threshold of 90% of the advertised maximum age (180ms) was used. The sixth variation used a threshold of 80% of the advertised maximum age (160ms). Table 5.1 summarizes the experiment groups and variations.

<table>
<thead>
<tr>
<th>Single Flow Topology Variations</th>
<th>Multiple Flow Topology Variations</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal DCCP</td>
<td>Normal DCCP</td>
</tr>
<tr>
<td>Basic Delete Mechanism</td>
<td>Basic Delete Mechanism</td>
</tr>
<tr>
<td>Pure ECN</td>
<td>Pure ECN</td>
</tr>
<tr>
<td>Advanced Delete Mechanism</td>
<td>Advanced Delete Mechanism</td>
</tr>
<tr>
<td>Hybrid Mechanism @ 90%</td>
<td>Hybrid Mechanism @ 90%</td>
</tr>
<tr>
<td>Hybrid Mechanism @ 80%</td>
<td>Hybrid Mechanism @ 80%</td>
</tr>
</tbody>
</table>

Table 5.1: List of topology groups and variations.

5.7.2.3 Experimental infrastructure components

The senders and receivers in the topology were configured to use the standard DCCP protocol as defined in RFC 4340 [4] at the transport layer. A simulated application layer was used to generate a high volumes of G7.11 [37] packets between the sender and the receiver pair (elastic flow not a simulated call). Congestion control was performed by CCID3 as defined in RFC4342 [8]. The IP protocol was configured as the network layer protocol.

Each link on the network was configured as a DS3 link operating at 44736 kbps. As can be seen in Figure 5.8, each sender was connected to the BOD1 device via a DS3 link. BOD1 then connects to an Internet cloud which is a simulation model device that acts as an advanced internet router and is where congestion on the network will occur. The Internet cloud connects to the BOD2 which finally connects to the receiver. (Note Figure 5.8 represents the multiple stream topology group. In the single flow topology group only one DCCP sender and receiver are present.)
The choice to use two BOD devices on the network was made to highlight the fact that multiple BODs can coexist along the same network path. Based on the topology however, only BOD2 will perform purging actions during the simulations. Purging actions were configured only to occur on DCCP-Data packets. All other packets are allowed to traverse the network unaffected.

5.7.2.4 Delay variation for the multiple flow topology group

In the multiple flow topologies, the delay on the network for each flow was changed numerous times to invoke three different scenario types in order to check for fairness amongst competing flows. In the first scenario type one flow was configured to generate older packets than the other two flows and therefore it experienced stale packets more rapidly. In the second scenario type, two flows were configured to generate older packets and in the third scenario type all three flows were configured to generate old packets. The sender does not generate older packets, instead the delay experienced before the packet reaches the BOD makes the packets older when they are received by the BOD.
Figure 5.9: Variation in the number of “old” flows used in the simulated topologies.
As can be seen from the topologies in Figure 5.9, delay was added to the flows in two places. The first delay was added to the flow between the server and the BOD1 device. As shown in Figure 5.9, the flows that generate stale packets are assigned a delay value of 150ms on this link. Other flows are assigned 100ms or 50ms of delay. The second delay value added to the flow is generated by the Internet cloud located between the two BOD devices. This delay value is random and varied between 40ms and 60ms. For the experimentation, the commonly accepted 200ms hard limit for VoIP communications was used. During the simulation, this 200ms limit was reached due to the combined 150ms delay and the random 40-60ms delay (when random delay value is >50ms). When a packet was found to possess a lifespan over 200ms the packet discard mechanism was configured to react accordingly. The effects these reactions had on that particular flow were then explored. In addition to the specific flow where stale packets were found, the effects the discarding had on the performance of the other flows was also monitored.

5.7.2.5 Application layer design

As in the previous experimentation described in Chapter 3 and 4, the application responsible for generating the traffic was not configured to behave as a typical voice/video application would be expected to behave. As the G7.11 codec was used to model the simulated application, the normal behavior for such an application would be to generate one data packet approximately every 20ms. In the experimentation described in this chapter, modeling the application to behave in such a way did not cause sufficient stress and congestion on the network. As a result when configured in a default way, the performance of the flow becomes a function of the 20ms limit found in G7.11 codec packets rather than a result of DCCP performance. For this reason, a decision was made to configure the application layer in such a way that it generated a constant stream of data packets. In doing so the DCCP layer and its associated performance was highlighted as this resulted in the DCCP send rate becoming the limiting factor in the performance of the flow. While doing this may not be realistic in the real world, it did highlight the weaknesses and strengths of using a discard mechanism in conjunction with DCCP.
5.7.2.6 Experimental assertions, assumptions and liberties

This section aims to describe any assumptions and liberties that were made/taken during the design of the simulation topology in order to provide as much rigor in the interpretation of results as possible. These are listed below:

- All end devices were configured with standard DCCP and CCID3 protocol stacks with the exception of additions described in Chapter 4 of this thesis.
- The experimental results were based on the fact that no congestion is experienced in the return path from the receiver back to the sender, in addition no acknowledgments were lost on this path either.
- All experiments use 200ms as the maximum age a packet is considered acceptable for. After 200ms the packet is deemed to be stale.
- All experiments use the same simulation clock in order to determine the current time when checking packet ages. The simulation clock was used in place of NTP which would be needed in real world topologies to provide this information.
- All data packets were 188 bytes in size as would be found in typical G7.11 voice streams (including deadline scheduling option fields).
- All other network traffic was switched off during the simulation. That is, no routing protocol information or updates were transferred during the simulation run period. Only data belonging to the relevant flows was present.
- No packets were dropped due to transmission errors or for any other reason apart from where defined in the results i.e. at the BOD.
- No delay was added to the overall network latency as a result of the BOD and its computations. In a real world implementation, computing time, queuing times and transmission times would occur as a result of implementing a BOD on the network. These factors were not taken into account in these results.

5.8 Experiment baseline topology worst case scenario

As previously described, the simulated application in this experimentation was configured to send as fast as the transport layer (DCCP) would allow. In addition, the queue on the intermediate internet device (router) was set to an infinite size. By configuring the network in such a way high, volumes of congestion on the network were created by a large queue size occurring at the intermediate device (router), which then led to stale packets occurring. This forced the various discard mechanisms to then take action in order to reduce the volume of stale packets. It is important to note that these conditions are not realistic. In reality, there would be limits on intermediate queue sizes and the application would likely be governed by a fixed packet rate in real time.
applications. This experiment although extreme was done to test the effectiveness of the theoretical discard mechanisms described in Section 5.4. The discard mechanisms should detect stale packets and take actions in order to reduce further stale packets from occurring. In particular BOD 2 is responsible for reducing the queue size found on the Internet node (router). This is where congestion is occurring and where stale packets emanate from due to long RTT times. By slowing the senders send rates to more appropriate levels, the BOD should reduce the RTT value and limit the number of future stale packets.

5.9 Single stream experiments

In this section the results of experiments carried out on the single stream topologies are presented. Following this the results obtained from the multiple stream experiment will be presented in Section 5.10.

These experiments aimed to quantify the effectiveness of the various discard mechanisms in terms of throughput, link utilization and goodput. In addition to this, a topology where none of these mechanisms were enabled was also tested. This was done so comparisons to the mechanism’s effectiveness could be made to standard DCCP behavior.

5.9.1 Throughput

Table 5.2 shows the number of packets that were sent in total, the number of packets that were dropped, marked or passed on unchanged (unmodified) and the calculated throughput rate per minute. The throughput rate represents the total number of packets that were received at the receiver for the minute testing period. The throughput rate does not take into consideration whether these packets were received stale or not stale.
Table 5.2: Throughput rates experienced during experimentation.

From the results table it can be seen that the normal DCCP topology that did not incorporate any discarding scheme was able to obtain the highest throughput rate of 49123 packets/min. The second highest throughput rate was achieved by the pure ECN topology which obtained a throughput rate of 48965 packets/min. This was 0.32% less than non-discard scheme. The third, fourth and fifth ranked throughput rates were achieved by the preventative hybrid mechanism (where ECN threshold value was set at 80% of the maximum age), the advanced delete mechanism and then again the preventative hybrid mechanism (where ECN threshold value was set at 90% of the maximum age) respectively. The model which produced the poorest throughput results was the basic discard mechanism which obtained a throughput rate of 44611 packets/min which is 9.19% lower than the optimal throughput rate obtained by the normal DCCP topology.

5.9.2 Goodput

The throughput results do not take into account the number of packets arriving that have become stale. By definition, the maximum age specified by the application when sending packets is the maximum time during which the packet remains useful to the application. If a packet is received with an age that is above this value, it is dropped by the application. Therefore having high throughput rates is not indicative of receiving higher numbers of useful packets. In Table 5.3, the goodput value is presented to better define how the models performed. In this table there are three variables presented. The first variable is the sent_stale variable. This value shows the number of packets that were forwarded out of BOD2 that were stale. This includes stale packets which were marked with the CE flag. The second variable was measured at the receiver and shows the percentage of packets that were received...
within their useful lifespan. This value is important as it shows the number of packets the application could use. The third value is the goodput rate of non stale packets received per minute in the simulations.

<table>
<thead>
<tr>
<th>Table 5.3: Goodput rates experienced during experimentation.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sent Stale</td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>Normal DCCP</td>
</tr>
<tr>
<td>Basic delete mechanism</td>
</tr>
<tr>
<td>Pure ECN</td>
</tr>
<tr>
<td>Advanced delete mechanism</td>
</tr>
<tr>
<td>Hybrid mechanism @ 90</td>
</tr>
<tr>
<td>Hybrid mechanism @ 80</td>
</tr>
</tbody>
</table>

These results shed an interesting light on what throughput values mean. Firstly, they show how many good packets are being received. Secondly, they can also be used to show how much bandwidth is wasted when stale packets are allowed to pass through the network. The topology configuration and application design were done in such a way as to cause maximum congestion and for the packets to very quickly exceed their useful lifespans if intervention on the BOD did not take place. These results must be viewed as potential yield from discard mechanisms in extreme conditions.

From the results it can be seen that in the normal DCCP topology where no intervention took place, only 2.17% of packets were received by the receiver within their useful lifespan period. Once the discard mechanisms were enabled on the same topology, the number of useful packets received all exceeded 99% with the exception of the basic delete mechanism. The basic delete mechanism resulted in 93.67% of packets arriving on time.

The best performing discard mechanism in terms of both percentage of packets arriving on time and actual number of non stale packets per minute was the preventative hybrid mechanism with an ECN threshold value set to 80% of the maximum TTL. The second best performing mechanism was again the preventative hybrid mechanism with an ECN threshold set to 90% of the maximum TTL. This indicates triggering the ECN marking phase to warn flows to slow down earlier, results in a higher volume and percentage of packets arriving on time.

The mechanism that was able to provide the third best performance was the pure ECN mechanism. While the advanced deletion mechanism resulted in a slightly higher
percentage of packets arriving on time when compared to the pure ECN topology (0.08% higher), the number of non stale packets sent per minute was significantly lower. The pure ECN topology was able to deliver 2975 more non stale packets in the same time period when compared to the advanced deletion mechanism.

5.9.3 Packet age

Another variable that was measured during the experimentation was the packet’s average age of the packets when they were received by the receiver. This variable was used to determine how fresh the packets were when they were received. A more desirable result is having younger packets being received rather than ones at the precipice of the maximum TTL value.

<table>
<thead>
<tr>
<th>Packet Age (sec)</th>
<th>Min</th>
<th>Max</th>
<th>Average</th>
<th>Std Dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal DCCP</td>
<td>0.001179737</td>
<td>13.36236763</td>
<td>6.62795725</td>
<td>3.886316679</td>
</tr>
<tr>
<td>Basic delete mechanism</td>
<td>0.0011798072</td>
<td>0.367328376</td>
<td>0.101377599</td>
<td>0.068050906</td>
</tr>
<tr>
<td>Pure ECN</td>
<td>0.001179505</td>
<td>0.279079378</td>
<td>0.082797701</td>
<td>0.032426205</td>
</tr>
<tr>
<td>Advanced delete mechanism</td>
<td>0.00117951</td>
<td>0.279079378</td>
<td>0.08257928</td>
<td>0.032143988</td>
</tr>
<tr>
<td>Hybrid mechanism @ 90</td>
<td>0.001179737</td>
<td>0.247393399</td>
<td>0.092028608</td>
<td>0.033709641</td>
</tr>
<tr>
<td>Hybrid mechanism @ 80</td>
<td>0.001179737</td>
<td>0.218856573</td>
<td>0.081928583</td>
<td>0.030138062</td>
</tr>
</tbody>
</table>

Table 5.4: Packet ages experienced during experimentation.

One of the concerns with the discard mechanisms was that they would result in an artificial throughput cap based on maximum packet age value for the flow. This could then lead to the average packet age becoming the maximum packet age due to this cap rather than a lower value. To explain the concern further, if the maximum age was set at 200ms which was the case in this experimentation, one of the concerns was the mechanism would result in the average age of packets becoming 200ms.

The results however show that this did not occur in the topologies that employed discard mechanisms as seen in Table 5.4. The act of reducing congestion and queue sizes performed by the discard mechanisms, coupled with the DCCP mechanisms for reducing send rates when drop or ECN events occurred, meant the packet average ages experienced were not artificially limited at the maximum 200ms. Instead, all of the discard mechanisms resulted in very healthy packet ages. The results show that the hybrid mechanism (ECN threshold set at 80% of the maximum age) resulted in a slightly lower average packet age than the other mechanisms. The second best
performer based on the results was the advanced deletion mechanism followed by the pure ECN mechanism.

Interestingly, the preventative hybrid mechanism with an ECN threshold set to 90% of maximum age performed less effectively than when the ECN threshold was set to 80% of the maximum age. There was a 10.1ms difference in the average packet ages received. This would again indicate a more aggressive ECN threshold of 80% of the maximum packet age will result in better performance. Of all the discard mechanisms, the basic deletion mechanism resulted in the least desirable performance. The average age of packets received by the basic deletion mechanism was 101.38ms. Having stated this however, it should still be noted that the average packet age was still acceptable at this level when compared to the topology with no discard mechanism.

5.9.4 Congestion levels

As per the findings in Chapter 3, an increase in queue depths leads to longer RTT values when the number of packets entering the queue exceeds the number of packets that can be transmitted (i.e. congestion). Increased levels in congestion lead directly to longer RTT times. In this experiment, there was a possibility of queuing taking place (congestion) at a number of points throughout the infrastructure. It was for this reason that the RTT value was used to measure the levels of congestion in the network rather than measuring individual queue depths. This was done as it provided a more cumulative value of the congestion levels on the network. As there was no contention for network resources in the acknowledgment path and no losses of acknowledgments, changes to the RTT value could only be caused by increased congestion levels. In Table 5.5, the minimum RTT, the maximum RTT and the average RTT experienced during the flow are reported. The standard deviation is used to show how much deviation from the mean occurred based on the results collected.
Table 5.5: RTT values experienced during experimentation.

<table>
<thead>
<tr>
<th>Method</th>
<th>Min</th>
<th>Max</th>
<th>Average</th>
<th>Std Dev</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal DCCP</td>
<td>0.0288</td>
<td>6.9268</td>
<td>1.110235714</td>
<td>1.909586814</td>
</tr>
<tr>
<td>Basic delete mechanism</td>
<td>0.0288</td>
<td>0.1306</td>
<td>0.084081619</td>
<td>0.025288684</td>
</tr>
<tr>
<td>Pure ECN</td>
<td>0.0214</td>
<td>0.1983</td>
<td>0.1114680027</td>
<td>0.052606814</td>
</tr>
<tr>
<td>Advanced delete mechanism</td>
<td>0.0175</td>
<td>0.1985</td>
<td>0.112908816</td>
<td>0.051744572</td>
</tr>
<tr>
<td>Hybrid mechanism @ 90</td>
<td>0.0205</td>
<td>0.1785</td>
<td>0.117832486</td>
<td>0.047617316</td>
</tr>
<tr>
<td>Hybrid mechanism @ 80</td>
<td>0.0166</td>
<td>0.1588</td>
<td>0.105522181</td>
<td>0.043029578</td>
</tr>
</tbody>
</table>

The minimum experienced RTT is of little consequence as it was the value of the first calculated RTT value. This abnormally low value occurs as there is no congestion at all on the network when the first packets are sent and subsequently acknowledged almost instantaneously, it is not representative of the mechanism.

In this experiment, the basic deletion mechanism was shown to maintain the lowest average congestion levels on the network when compared to the other discard mechanisms. This was found to be caused by the continual dropping of packets which never allowed the senders rate to take full advantage of the available bandwidth. Throughout the flow, there were periods where link utilization levels dropped to below 100% as there were periods where the transmission rate exceeded the number of packets being queued. This means that the basic deletion mechanism was over active in preventing congestion from occurring and as a result did not utilize the resources available efficiently. This is supported by the evidence in throughput and goodput rates where the deletion mechanism was found to result in much lower throughput and goodput rates. In all the other discard mechanisms experiments, at no point after reaching the 100% utilization rate did any of the topology links cease to operate at full capacity. Fully utilizing available resources is more desirable but does lead to higher levels of congestion. This fact prevents the basic deletion mechanism from being considered the best mechanism at lowering congestion levels on the network.

Therefore the preventative hybrid mechanism which utilized an ECN threshold of 80% of the maximum TTL resulted in the lowest average RTT values. Again setting the ECN threshold at a more aggressive 80% rather than 90% of the maximum age resulted in better performance of the mechanism. The advanced deletion, the preventative hybrid (set at 90%) method, and the pure ECN all resulted in similar RTT values.

When compared to the standard DCCP topology where no discard mechanism was
present, all the models performed more efficiently at reducing the levels of congestion on the network. The preventative hybrid mechanism (ECN threshold set to 80% of the maximum age) has an average RTT value that is at least 10ms less than the other discard mechanisms were able to produce making it the mechanism of choice for reducing congestion levels.

5.10 Multiple stream experiments

The reason for configuring the network to produce multiple simultaneous flows/streams was to allow the interactions between the multiple flows to be examined with the main focus of these interactions being to investigate fairness between flows. Defining exactly what fairness in DCCP constitutes when multiple flows exist proved to be quite difficult. Fairness in the context of this research pertains to an equitable distribution and/or allocation of resources to all network flows wishing to utilize the shared resources.

The problem of applying fairness rules to DCCP flows to measure levels of fairness is two-fold. Probably the most widely used work on determining fairness in communication networks today originates from the original work carried out in [7]. In this work, Jain states: “There is little agreement amongst researchers as to what should be equalized. In computer networks, some want equal delay, others want equal throughput, and yet others want equal power for all users sharing a resource” [7]. This extract summarizes the first problem in relation to fairness determination in DCCP and CCID3. That is, what elements should be used to determine fairness? Is fairness having each flow being granted access to exactly the same bandwidth allocation? Is it having each flow being exposed to the same amount of delay etc...? While mathematical formulas are effective in theorizing how this perfect fairness can be achieved and exactly which metrics are best at doing this, it is extremely difficult to accept that any theory based mathematical equation would be able to fully take into account all of the nuances of complex mechanisms like DCCP and CCID3. Issues like which phase the congestion algorithm is in, queue depths, number of competing flows and a multitude of other variables must be considered.

The second problem with the notion of fairness is that in order to provide absolute fairness in complex networks, there is a commonly overlooked point in that any at-
tempt at implementing a fairness mechanism will rely on after-the-event reactions in order to calculate the fairness values. Given the implementation can only rely on past events to try and predict future ones means there is no guarantee of absolute fairness if there is any randomness in what can occur after these events. A simple example to highlight this is that a device cannot know how many simultaneous flows it can expect to occur in the future. It can only assign fair values based on the currently known number of flows. The reality of the matter is that absolute fairness is only achievable if all the future events can be predicted precisely. This is not the case in a complex DCCP CCID3 flow or on a network. If one cannot implement a fairness scheme, then absolute fairness really can not be expected in the first place.

In this chapter fairness is measured in terms of “less fair” relative to the baseline fair model which will be described below. Instead of trying to prove that the various discard mechanisms are “fair”, this research instead focuses on areas in the results that clearly identify when actions have led to the existence of unfairness.

5.11 Baseline unfair topology

To determine if there was an increase in fairness or a decrease in fairness as a result of the implementation of a discard scheme, a baseline experimental topology used for all testing was created. In this baseline or normal DCCP topology no discard schemes were utilized in the topology. From this baseline topology the following observations were made to show that the topology created was indeed unfair. If all the streams were initiated at exactly the same time, then the first stream to receive an acknowledgment was able to obtain a significantly larger proportion of the available bandwidth. As a result, the two other flows which were present received much lower proportions of the bandwidth. These results are demonstrated below in Figure 5.10.
Figure 5.10: Baseline fairness model used to measure fairness. *No DCCP discard mechanism is enabled.*

Analysis of the data shown graphically in Figure 5.10 shows that Flow 1 originating from server_dccp_1 received the first acknowledgment. As the network is intentionally configured to be unfair, flow one was able to obtain an unfair proportion of the bandwidth as its congestion window increased faster than the other two flows. The rapid growth of Flow 1’s congestion window limited the other two flows from growing as quickly. The results also showed that Flow 3 originating from server_dccp_3 was the second device to receive an acknowledgment and was able to start slow start phase before Flow 2. This led to a situation where Flow 3 was able to gain a larger proportion of bandwidth when compared to Flow 2. To ensure this was the cause, a delay of 50ms was added to the network link between server_dccp_1 and the BOD1. The experiment was then run again. The send rate window reported for the three streams is presented in Figure 5.11.
Again analysis of the data was carried out and the results showed that the first acknowledgments received were in the following order. The first acknowledgment received was for Flow 3 (server_dccp_3) followed by Flow 2 (server_dccp_2) and finally by the delayed Flow 1 (server_dccp_1). As can be seen in Figure 5.11, this order relates to the proportion of bandwidth obtained by the flow for the remainder of the session. This topology was deliberately created in such a manner in order to showcase clear network bandwidth allocation unfairness between flows and to see if the discard mechanisms were able to rectify these issues or if they would in fact contribute to higher levels of unfairness.

Having created a base topology that represented a scenario where the first flow to initiate became the dominant one for the remainder of the session, the discard mechanisms were then enabled on BOD2 in the topology and the same simulation run. Regardless of bandwidth allocation and proportioning, the network is designed to become congested during the simulation triggering the BOD discard mechanisms to start discarding stale packets that result as a consequence of this congestion. The previous experiments show that the discard mechanisms are effective at reducing the sender’s send rate when congestion levels in the network increase. As a result of
this reduction, it was believed the actions of the discard mechanism would result in one of two possible outcomes. Firstly and the least desirable outcome, was that the less dominant flows would be more severely affected by the actions of reducing the send rate caused by the BOD when stale packets were dropped or marked. As the dominant flow has a higher send rate, it would also be affected, but less so than the less dominant flows. This is because it is able to absorb the loss events more easily due to its higher transmission rate. When the less dominant flows reduce their send rates, the overall congestion rate would lessen in the network which would allow the dominant flow to take up even more of a disproportionate amount of the total available bandwidth on the network.

The second possible outcome as a result of the discard mechanisms action would be that once the congestion levels in a network reached a certain threshold, all flows would generate stale packets prompting a reduction in the send rates for all flows. The effects of the loss events would be more dramatic on the dominant flow as more simultaneous loss events would occur when congestion reached the BOD threshold. This is because the dominant flow would be sending a much higher volume of data across the network that becomes stale as a result of the congestion. This would mean it experiences more loss or marked packets and would have a more radical drop in send rate as a result of the increase in its P Value (loss rate). This relieves the dominant flow of some of its bandwidth which can then be utilized by the other less dominant flows making the distribution of bandwidth more proportional amongst streams.

With both outcomes equally probable, the experimentation tested whether the discard mechanisms would promote fairness or add to existing unfairness in the already unfair topology. In the first set of experiments, a discard mechanism was applied to all the flows in the unfair topology. In the second set of experiments, the discard mechanism was only applied to certain flows in the unfair topology. This was to test how a network functions if some of the flows are controlled by discard mechanisms and other flows operate without a discard mechanism (i.e. normal DCCP with no intervention).
5.11.1 Fairness between flows using the same discard technique

In this set of experimentation, each packet discard mechanism was applied to the multiple stream topology described in Section 5.2. As explained, three variations of these simulations were then run on each of the mechanisms tested. The first variation was a network in which one older flow was present. In the second variation, two older flows were present and in the third variation, all three flows were initiated at the same instant (no flow was old).

5.11.1.1 Fairness amongst streams initiated at the same time

First fairness amongst flows when the flows are initiated at exactly the same time was tested. The results shown in Figures 5.12 to 5.15 illustrate the send window rates that were experienced by each of the streams in the experiment.

![Graph showing send window rates](image)

Figure 5.12: Results used to measure fairness amongst streams initiated at the same time. Basic deletion mechanism active on the BOD.
Figure 5.13: Results used to measure fairness amongst streams initiated at the same time. Pure ECN mechanism active on the BOD.

Figure 5.14: Results used to measure fairness amongst streams initiated at the same time. Advanced delete mechanism active on the BOD.
Figure 5.15: Results used to measure fairness amongst streams initiated at the same time. Hybrid discard mechanism active on the BOD.

Figure 5.12 shows the send rates experienced when the basic deletion mechanism was activated. Figure 5.13 shows the send rates recorded when the pure ECN method was used. Figures 5.14 and 5.15 show the send rates measured for the advanced deletion mechanism and the preventative hybrid mechanism respectively.

From the results, it is clearly visible that when all of the packet discard mechanisms were enabled, there was noticeable improvement in fairness of bandwidth utilization on the network when the results are compared to the baseline unfair topology (see Figure 5.11). It is clear in all the graphs, that at the beginning of the simulation, one flow quickly utilizes all of the bandwidth available. However, what differentiates the results from the discard scenarios and the baseline unfair topology is that this individually dominant rate quickly lowers when congestion occurs. Once this drop occurs as a result of stale packets being removed from the network, the other two flows obtain a more proportional allocation of the total bandwidth. In terms of providing “more fair” results, it appears the basic deletion mechanism provides the fairest mechanism in the experiment. This was followed by the pure ECN model, the hybrid preventative model and then the advanced deletion mechanism respectively.

One clear assertion that can be made from these results is that all of the discard mechanisms made the allocation of bandwidth between streams appear “more fair”.

138
At no stage in the discard scenarios was one flow as blatantly deprived of bandwidth when compared to the scenario where no discard mechanism was used.

5.11.1.2 Fairness between streams starting at different times

In this experimentation, a slight variation of the above experimentation was conducted. In this set of experiments, the fairness between flows with different throughput characteristics was tested. In particular, the experimentation was conducted to determine if the discard mechanisms lead to scenarios where older flows are disadvantaged because of the discard mechanism's actions. In the first iteration of the experiment, Flow 1 was configured to have an additional 50ms delay in the network path between sender and BOD1. The other two flows did not experience this delay. In the second iteration, the 50ms delay was added to the second flow in addition to the first flow. The third flow was not impacted by this delay. Figures 5.16 to 5.20 show the send rates that were reported by the sender in each of these iterations (i.e. one old flow and two old flows).
Figure 5.16: Fairness between streams starting at different times. No Discard mechanism active i.e. Normal DCCP operation.
Figure 5.17: Fairness between streams starting at different times. Basic delete mechanism active.
Figure 5.18: Fairness between streams starting at different times. Pure ECN mechanism active.
Figure 5.19: Fairness between streams starting at different times. Advanced deletion mechanism active.
5.11.1.3 Discussion

The results show that in all of the topologies where the discard mechanisms are activated, a fairness issue is created when flows with different network requirements exist. The results from the baseline topology, where there was no discard mechanism enabled, show that the older flow[s] in the original topology did not timeout in the
baseline model as they do in all the topologies where one of the discard mechanisms was activated. This would suggest that where flows with different throughput rates exist, the use of a discard mechanism will cause an increase in unfairness in the proportion of bandwidth the older flow is able to obtain. In the case of this experimentation, this level of unfairness led to the collapse of the older flows entirely.

The difference between the first variation of the experiment where the discard mechanisms were seen to promoted fairness and this variation where the opposite is believed to be true, is that all the flows in the original topology were initiated at the same time and all required similar throughput rates. In this variation, the delay added to the old flows makes the flow less competitive and meant the packets became stale faster when congestion occurred. It was also discovered that the additional 50ms delay meant that the send rate was also limited. As shown in the baseline topology results (Figure 5.16), this delay was found to cause a reduction in the overall throughput rate of that flow. While it was to be expected based on CCID3’s use of RTT in calculating send rate, the baseline model shows this alone is not enough to cause the older flows to timeout. This process of elimination shows that the actions taken by the discard mechanisms caused the unfairness in the topology that led to all of the old flows timing out before the simulations completed.

In depth analysis of why this occurred found three contributing factors that led to the total starvation and eventual collapse of the old flows during the experiments. In the first variation where all flows had the same delay in the network it was shown that when a particular flow grew disproportionately large and congestion occurred, the dominant flow experienced a much larger number of drops compared to the less dominant flows. This meant the dominant flows send rate was reduced more rapidly, than that of the other less dominant flows. At the moment when the dominant flow’s rate was dropped, the less dominant flows were in a more favorable position to take up the bandwidth that the dominant flow lost as a result of the drop actions. In this variation of the experimentation where 50ms delay was applied to the less dominant flows, these flows were not as aggressive in trying to utilize this bandwidth that becomes available. Through lower delay, the dominant flow is able to more rapidly reclaim any of the dominance it loses as a result of its send rate drop. In addition, when congestion occurs and the less dominant flows generate stale packets, their rates are also reduced. When this occurs, the more aggressive dominant flow is able to take
control of this bandwidth which leads to the eventual starvation of all bandwidth from the less dominant flows as is seen in Figures 2.17 to 2.20 where the older flows timeout.

While the aggressiveness of obtaining released resources was the main cause of the unfairness seen when discard mechanisms were enabled, two other contributing factors should also be mentioned. In the first variation of the experiment where there was no variance in delay, when congestion occurred the dominant flow was found to have experienced a much larger number of packet drops as a result of congestion when compared to the less dominant flows. This is because when the queue size grew to levels where packets became stale, as a result of waiting to be transmitted, the majority of these packets belonged to the dominant flow. The 50ms delay added to the older flows resulted in a much higher number of drops occurring in the less dominant flows when compared to initial variation of the experiment. With these flows experiencing higher levels in drop rates, they were less well positioned to capitalize on any of the bandwidth seized from the dominant flow.

The final finding, which relates largely to the above mentioned one, is that the 50ms delay meant that the older flows packets became stale at a faster rate than the normal flow. It was found that when the queue levels grew to a point where they were adding 150ms of delay to the packet’s life through delay, the less dominant flows started to experience stale packets whereas the more dominant one did not until after 200ms, as it was not affected by this delay. This led to the less dominant queues lowering their send rates before the more dominant one was required to do so. Being more aggressive, the more dominant flow was able to utilize any freed resource that resulted from this, causing more starvation to the less dominant flows.

These findings show that that when multiple flows with different characteristics traverse the network, the DCCP discard mechanisms can add to unfairness. It should be noted that the topology used for experimentation is extreme and not indicative of what would be found in the real world.
5.11.2 Fairness between standard DCCP streams and those controlled by discard mechanisms

The experiments conducted above have shown that the old flow in the experimental topologies is starved of bandwidth and collapses as a result of unfairness. These experiments focus specifically on whether a flow that experiences no discard mechanisms actions will starve the bandwidth of an identical flow (in terms of delay) which has actions taken on it by a discard mechanism. In this set of experiments, one of the three flows was allowed to traverse the network without being affected by the delete mechanism. The second flow was configured to be old (i.e. added delay was placed in the flow’s path) and was subjected to the delete mechanism. The third flow was configured to be normal (not old) and was configured to experience the same delay as the standard DCCP flow where no discard was used. The results obtained from these experiments are shown in Figures 5.21 to 5.24. As in all of the above experiments, the send rates reported by sender are used to show fairness of bandwidth allocation.

Figure 5.21: Send window rates for streams used to demonstrate fairness amongst dissimilar stream types. Basic delete mechanism active.
Figure 5.22: Send window rates for streams used to demonstrate fairness amongst dissimilar stream types. Pure ECN mechanism active.

Figure 5.23: Send window rates for streams used to demonstrate fairness amongst dissimilar stream types. Advanced delete mechanism active.
In the results shown in Figures 5.21 to 5.24, Flows 2 (red lines) and Flows 3 (green lines) are of major interest. Flow 2 represents a normal DCCP flow where no discard mechanism actions occur while Flow 3 represents a normal DCCP flow upon which discard mechanisms actions take place. From the results, it is clear that in all of the simulations with the exception of the pure ECN model, all of the DCCP flows where discard mechanisms actions take place collapse as a result of unfairness caused by the actions taken by the discard mechanism. The pure ECN flow where ECN marking takes place but no dropping occurs is able to compete fairly with the normal DCCP flow. This suggests that the action of dropping packets results in the unfairness. The pure ECN mechanism was found to offer exceptionally good fairness levels where the flows were equally competitive and appears suitable to mixed transport layer networks.

5.12 Conclusion

Having described the benefits of embedding packet life and how this could be achieved in DCCP in Chapter 4, this chapter introduced five discard schemes which take advantage of this added feature. Having described these mechanisms and how they would
be implemented, as well as some of their respective advantages and disadvantages, the chapter then tested through proof-of-concept experimentation if these mechanisms were able to improve DCCP performance in severely congested networks.

From the results obtained from the experimentation, it was found that all of the discard mechanisms that were tested in the experimentation were shown to greatly improve the goodput rates and lead to lower average packet ages being received at the receiver. In addition, all of the discard mechanisms tested were able to reduce the congestion levels that occurred in the network.

Further experimentation revealed however that while the discard mechanisms promote fairness when multiple streams possess exactly the same throughput requirements and initiate at similar times, they cause fairness issues when the streams do not have the same requirements or where some of the streams are not controlled by a discard mechanism and others are. The one exception to this however was the pure ECN mechanism.

Probably the most significant finding in this research is that the pure ECN mechanism, which does not drop any packets, was able to provide fairness where there were combinations of non-controlled flows or flows with different throughput requirements. While the pure ECN model was not shown to offer the best goodput or lowest average packet ages when compared to the other discard techniques, it did perform significantly better than the baseline model, which did not employ any discard mechanism. For this reason, the pure ECN model is deemed a workable solution that could be implemented in its current form onto DCCP networks.

The other discard mechanisms described show much promise for providing more timely delivery of real time data to the application, but will need to overcome the fairness issues they create before they could be adopted on multi-transport layer networks. This chapter has shown adding the packet life information into the DCCP header creates a number of new possibilities and ways in which more efficient delivery of real time data transportation can now occur.
Chapter 6

A novel probabilistic based scheduling algorithm for DCCP

6.1 Introduction

This chapter will introduce a novel probabilistic based scheduling (PBS) mechanism designed specifically for use with the deadline enabled DCCP protocol. This PBS mechanism shows another potential application for the option fields appended to the DCCP header as described in Chapter 4 in order to make transfers of real time data more efficient. To achieve this improvement, routing metric information is used to predict the time needed for a packet to reach its intended destination network. Based on this estimate scheduling of packets in queues on the device is optimized. This chapter is broken into two parts. The first part describes the PBS mechanism at a theoretical level and discusses the design and implementation of the mechanism. In the second part experimentation undertaken to test the reliability, scalability and accuracy of the PBS mechanism as well as the performance gains the PBS mechanism produced, is described.
6.2 Overview of the PBS mechanism

In previous chapters (Chapters 1-5) this thesis has demonstrated how real time data possesses a finite lifespan and described a mechanism that allows this lifespan property to be incorporated into the transport layer in order to allow more efficient transportation of such data. In this chapter, the variables added through the deadline scheduling mechanism will again be utilized to procure more efficient delivery of real time data across a network. This is to further validate the merit of incorporating deadline scheduling into DCCP by showcasing one of many ways in which this inclusion is useful. This PBS mechanism will expand upon the principles of the discard mechanisms described in Chapter 5. Instead of dealing purely with stale packets, the PBS mechanism described in this chapter will calculate a packet’s probability of arriving at its destination within its useful lifespan to determine if it should be sent or not and the priority needed to send it. In order to do this, information relating to the network located between the device where the forwarding decision is made and the end point is derived from a routing protocol. This essentially means three layers of the OSI model are utilized in the transportation decision making process.

6.2.1 High level overview of the PBS mechanism

As is described in Chapter 4, the application layer on a deadline scheduling enabled DCCP sending device passes down the packet’s maximum lifespan duration value concurrently with the passing down of the payload data. Upon receiving this, the DCCP layer on the sending device timestamps the packet with a birthtime variable. This birthtime is extracted from the NTP synchronized system clock and represents the time the packet was generated. The payload data is then encapsulated into a generic DCCP-Data packet with the two above-mentioned variables appended into the lifespan option fields. The packet is then passed to the relevant network layer protocol for transportation across the network. Located within the network between the sender and the receiver will be a number of intermediate devices known as BODs. These devices are modified/enhanced layer 3 and above network devices such as routers and layer 3 switches (switched networks). In order to convert these devices into BOD’s the PBS mechanism described in this chapter or the mechanisms described
in Chapter 4 are added to the device to allow it to utilize the lifespan variables and offer improved transportation of real-time data. In this PBS mechanism, when the BOD receives a packet, it also queries/extracts information from the routing protocol in order to determine the approximate time it will take for the packet to reach the destination. Once this time has been estimated, the PBS mechanism will then determine if the packet has sufficient lifetime remaining to reach the destination. Using an algorithm which will be described in detail in Section 6.4.1.1, packets will be assigned a probability of arriving on time and categorized into four distinct groups. This allows more advanced scheduling decisions to be made in order to offer the greatest good to all packets attempting to traverse the network. In addition packets that have no possibility of arriving within their useful lifespan are also removed from the network making additional bandwidth available to other flows which would be impacted by the transmission of these redundant packets. Before delving into the specifics of the PBS mechanism, an overview of related research is presented in Section 6.3.

6.3 Motivation and related research

The primary motivation behind the creation of this PBS mechanism was derived from [29]. In this work, Cheng and Leung describe a novel mechanism for discarding packets that are likely to expire, in order to reduce queuing periods on wireless networks. In doing this, they suggest fewer expired packets will result due to the shorter queuing time needed while waiting to be transmitted. Their work is based on research found in [50, 51] which demonstrates the relationships between queue sizes and average packet ages on wireless networks. Having described the potential for determining the probability of a packet expiring before reaching its destination, Cheng and Leung determine the process is computationally exorbitant and instead implement a gentle implementation of RED to control the size of the queue. In both the mathematical modeling and simulation they perform, the merits of controlling the queue size is demonstrated. The PBS mechanism in this chapter is based loosely on their original idea but varies significantly from their work. Firstly Cheng and Leung’s approach is performed at the data link and physical layer [29]. Their mechanism is closely cou-
pled with the wireless mechanisms on the data link layer and physical layer whereas, the PBS mechanism described in this research operates at the transport layer. From [29] it becomes apparent the task of computing a packet’s probability of arriving at the destination within its useful lifespan is somewhat difficult at the data link and physical layer and as a result, they abandon the idea and instead make use a RED implementation to control queue sizes. In order to overcome the complexity experienced by [29] in trying to calculate the probability of the packet arriving within its lifespan, this scheme moves the mechanism to the transport layer and accesses information from a routing protocol at the network layer. By utilizing network information from the routing protocol to determine the network characteristics, the task of calculating the probability a packet has of arriving at its intended destination within its useful lifespan becomes possible as will be shown. This solution provides a more scalable solution to queue management when compared to the one presented in [29]. As DCCP does not retransmit lost packets, the act of dropping packets has a different effect on the flow when compared to TCP used in [29]. Finally this research is designed specifically for the nuances of DCCP and is network layer independent which makes it unique.

While the idea of purging packets based on their probability of being received on time originates from [29], when it came to actual implementation of this PBS mechanism a loose connection between this mechanism and a low latency queuing mechanism [52] was formed. As the goal of low latency queuing is to reduce jitter and delay for time sensitive network flows, its aims are very similar to that of this PBS mechanism. For this reason some inspiration was drawn from LLQ during the design of this PBS mechanism. In particular the LLQ model makes use of numerous queues with guaranteed bandwidth allocations and fixed service rates, which is similar to the queuing and scheduling techniques that are used in this PBS mechanism. This mechanism does however differ significantly from LLQ as instead of categorizing packets based on tags/markings as occurs in LLQ, this PBS mechanism categorizes packets based on their ages and probability of arriving within their lifetime.
6.4 Definition of terms and acronyms

In the following sections a number of acronyms will be used in order to keep the specification concise. In this section these acronyms will now be defined with an accompanying description of where they are derived from. The first group of acronyms are derived directly from the Chapter 4 specification. The acronym BT will be used in place of birth time. This value is created when DCCP at the sender receives data from the application layer and encapsulates the payload into a DCCP-Data packet. This value is obtained from the system clock and is the exact time at which the DCCP packet comes into existence. When the application layer passes the information down to the DCCP layer, it also passes down a maximum time to live (MTTL) variable. This value represents in milliseconds the amount of time the real time data is valid for. In order to determine if a packet has expired or become stale, the MTTL value is added to the BT value to get the explicit NTP expiry time (ENET). At any point in the transmission an intermediate device can query the NTP clock on the system for the current NTP time (CNT). If the ENET > CNT then the packet contents are stale. In order to find out how much life time a packet has remaining the time left to live (TLTL) variable is used. To calculate the TLTL the ENET is subtracted from the CNT (TLTL = ENET-CNT).

In addition to the above terms and acronyms, this PBS mechanism utilizes two other variables. The first of these variables is known as time left in queue (TLIQ). This variable is calculated using the formula described in Chapter 3 Section 3.6.1, and represents the minimum amount of time it will take for a packet to be transmitted in the current queue if no change is made to the position of any packets in that queue. The second new variable is known as time needed to cross network (TNTCN) variable. This value is derived from the routing protocol and represents the estimated time that will be needed for the given packet to be delivered from the current device to the destination network based on current network conditions. How the TLIQ and TNTCN variables are calculated will now be presented in greater detail.
6.4.1 Time left in queue (TLIQ)

From the moment the PBS mechanism described in this chapter is initiated, it is vital that queue depth information is monitored and updated constantly for each of the queue categories. In addition to this the current expected queue delay for newly arriving packets must also be calculated continuously. To achieve this the TLIQ variable is calculated every time a packet is received by the BOD device. This TLIQ value represents the time a packet will remain on the device while in the outgoing interface queue. The TLIQ variable is critical in determining and differentiating whether a packet should be placed in the critical queue or the normal queue. The purpose of these queue types is described in Section 6.5.1.1. The formula for calculating the TLIQ value is as follows:

\[
TLIQ = \frac{TPS}{TX \times BW\%} \tag{6.1}
\]

In order to calculate the TLIQ value for a newly arriving packet, the following steps take place. First the interface service rate is calculated by taking the interface speed in kilobits (TX) and multiplying it by the bandwidth percentage (BW\%) available to that queue. Next the queue depth information is accessed to determine the cumulative size of all existing packets in the queue in kilobits (TPS). From these two variables, the time that will be required for the queue to be serviced is calculated. As previously mentioned, the TLIQ value must continually be calculated to ensure it is always up to date. When a new packet arrives, the TLIQ value must be readily available to the PBS mechanism so it can determine whether the packet belongs in the critical queue or the normal queue.

6.4.1.1 Time needed to cross network (TNTCN)

The novel element of this PBS mechanism is provided through the addition of the TNTCN variable. This variable is derived using information obtained from a routing protocol and is used to provide an estimate of the time that will be required for the packet to reach the destination network. Using this value the probability of the packet
arriving at the intended destination within its useful lifespan can be calculated. In this research Cisco System’s Extended Interior Gateway Routing Protocol (EIGRP)\[53\] is used to calculate the TNTCN variable. There are a number of reasons why this routing protocol was chosen. Firstly the EIGRP protocol uses a number of variables to calculate its routing and topology tables. These variables include delay, reliability, load, bandwidth, hop count and MTU \[54\]. Each of these variables represents unique conditions that exist in the network between the router’s queue and the destination network. The aim behind this research is to promote cross layer cohesion and to reduce computational overheads and task repetition wherever possible. To configure this PBS mechanism to calculate the variables which EIGRP already provides was deemed redundant when these variables are already readily available. To add to this, the algorithms used to derive these variables in EIGRP are well matured and have been proven to work efficiently. The second reason why EIGRP was used, is that compared to other routing protocols, EIGRP, through a series of configurable metrics, is highly customizable. Borrowing from EIGRP’s customizable design, this PBS mechanism also allows the TNTCN value to be derived from any combination of the variables used in EIGRP. This ensures that the PBS mechanism can be configured in a way that best suits network conditions. For example, the algorithm could be configured to use only EIGRP’s reported delay variable when calculating the TNTCN value. In scenarios where reliability and bandwidth are concerns, these values can be factored into the TNTCN probability calculation.

The final reason why the EIGRP routing protocol was selected was because it offers fast convergence times through its Diffusing Update Algorithm (DUAL). Fast convergence times equate to higher levels of accuracy in relation to actual network conditions, which is a prerequisite for this PBS mechanism. If changes to the network occur the routing protocol must be able to detect these quickly in order to provide accurate information to this PBS mechanism. Failure to do so in a timely manner will lead to incorrect scheduling and prioritization of packets.

The drawback to using the EIGRP protocol is that it limits this PBS mechanism purely to topologies that are configured with CISCO devices. In addition to this, EIGRP is an interior routing protocol and therefore this limits the scale of the PBS mechanism to single autonomous systems and not the broader Internet in its current form. Finally, it is also worth mentioning that EIGRP does utilize bandwidth in
order to communicate routing updates to other routing devices on the network. This research will assume that the EIGRP protocol has been configured on the network device for core routing functionality and not for the express purposes of this PBS mechanism. Therefore any additional overhead needed for EIGRP convergence such as hello packets will occur irrespective of the implementation of the PBS mechanism. There is no reason why a different non-proprietary or external autonomous system routing protocol could not be substituted into this PBS mechanism if suitable.

6.5 PBS mechanism details

There are two main elements that make up the PBS mechanism. Firstly, there is the categorization element that is used to determine which category and subsequently which queue is appropriate for an incoming packet that is scheduled to be forwarded out the interface. The second element is a packet scheduler which services the respective queues in such a way as to ensure strict adherence to the queues allocated bandwidth percentage. These elements will now be discussed.

6.5.1 Categorization of packets

6.5.1.1 Queue structure

The first phase of the PBS mechanism involves the categorization of incoming packets into three unique queues based on information gathered from the BT and MTTL variable in the DCCP header option fields. This section will commence by describing the function/role of each of the three queue classes.

Discard queue

The first packet queue class is for packets that have expired or will do so before they reach their intended destination network. This queue, known as the discard queue, will employ techniques such as those described in Chapter 5 to ensure there are not excessive packet drop events and that the session remains active. In order to qualify for this queue, the following two criteria are checked. Firstly, if the packet arrives stale \((ENET < CNT)\) then the packet is placed in the discard queue. Secondly,
if the packet will not reach its destination before becoming stale \((\text{TTL}_L < \text{TNT}_C N)\) then the packet will also be placed into the discard queue.

**Critical queue**

The second queue class is named the critical queue and is for packets that will expire unless a prioritization action takes place to prevent them from doing so. Specifically, if the process of passing through the queue on the BOD will directly lead to the packet becoming stale then the packet is placed into this queue. The critical queue is allocated a predetermined amount of bandwidth in order to handle such events. To qualify for this queue a packet must meet the following criteria. The packet must have sufficient lifespan remaining to allow it to traverse the network to its intended destination without becoming stale \((\text{TTL}_L > \text{TNT}_C N)\). In addition to qualify for this queue the packet must be in a position where the accumulated time needed for queuing \((\text{TLIQ})\) and the time needed for the packet to cross the network is greater than the time the packet has remaining before becoming stale \((\text{TTL}_L)\). To summarize, the following formula applies: \(\text{TTL}_L > \text{TNT}_C N \text{ AND } \text{TTL}_L < (\text{TNC}_N + \text{TLIQ})\). The logic behind this queue is that packets that can reach the destination, if they are not delayed excessively by the queuing process on the device, are given the best chance to do so by being placed in a smaller more rapidly serviced queue. If this action does not occur, the packets will become stale before reaching their destination as a result of queuing.

**Normal queue**

The final queue class is for packets that will likely reach the destination network within their respective lifespan provided there are no abnormal network fluctuations or extreme changes in network conditions. To qualify for this queue the packet must possess a lifespan \((\text{TL}_L)\) greater than the time needed for queuing and the time needed for the packet to traverse the network to the destination device. i.e. \(\text{TL}_L > (\text{TLIQ} + \text{TNT}_C N)\). Packets in this queue will be offered a guaranteed position and guaranteed bandwidth allocation ensuring the queue time is deterministic. Pack-
ets placed into this queue are placed there on a first in first out (FIFO) basis and no packet is ever to be placed in front of a pre-existing packet. This ensures packet queuing times become deterministic. As this queue is given a majority share of available bandwidth and offers guaranteed service rates, it creates an incentive for the application specifying the MTTL values to actively aim to send packets that are placed into this queue. In the flow diagram in Figure 6.1, the categorization mechanism described above is demonstrated graphically.

![Flow diagram showing categorization of packets logic.](image)

### 6.5.2 Determining the packets probability of arriving in time

When a packet is received for transmission on an interface, the algorithm parses the packet’s DCCP and IP header to obtain three variables. The first two variables that are extracted are derived from the additional deadline scheduling lifetime fields described in Chapter 4 Section 4.2.2.4, namely the packet’s BT and the packet’s MTTL. In addition to this the IP header is also accessed to determine the packet’s intended destination network. Finally the routing protocol information is accessed in order to estimate the forward path delay the packet can expect to experience using EIGRP metrics. Once this value is established, the PBS mechanism is able to predict the probability of a given packet arriving at its destination within its useful lifespan.
6.5.2.1 Using EIGRP to calculate TNTCN

In order to determine the delay between the BOD and the destination network there are three elements that make up the TNTCN latency value. The first of these elements is commonly referred to as wire line latency. This value is the time it takes for the data signal to travel across wired or wireless medium between intermediate and end point devices including the serialization of packets into the necessary electrical signal format. This latency is governed by principals of physics and is almost always deterministic. Typically compared to the other two types of latency that will be discussed next, this value is negligible in size (except radio and ADSL links).

The second element that contributes to the TNTCN value will be referred to herein as switching latency. This value represents the minimum latency that is added by the intermediate devices if no queuing takes place. This latency is created by the internal activities that occur on intermediate devices such as switches and routers with the exception of any queuing related activities. Examples of these activities include transferring packets between interfaces, packet encapsulation, MAC table and routing table lookup activities to name but a few. These values are also all typically deterministic and will normally remain constant throughout a flow.

The third component that contributes to the TNTCN value is queuing latency. Although queuing takes place on intermediate devices, this value will be treated separately in this research from the switching latency variable described above. The reason for this is that incorporation of queues into the network path introduces a non-deterministic element into the delay calculation. As queue sizes fluctuate throughout a session, this causes variance in latency values the various flows traversing the network will experience. In order to provide an accurate TNTCN variable to the PBS mechanism, all three of these values must be taken into consideration. The TNTCN variable is therefore calculated as shown in Equation 6.2.

\[
TNTCN = SwitchingLatency + WireLatency + QueuingLatency
\]  

(6.2)

Having defined the three components that make up the TNTCN variable, the way in which the EIGRP metrics are used to calculate these elements will now be
described. To estimate the time that will be required for the packet to travel across the remaining portion of the network, four of EIGRP's metrics are used by the PBS mechanism. These four metrics are EIGRP delay, load, bandwidth and hop count. Specifically, the delay value from EIGRP will be used to calculate the wire latency value component of the total expected latency. The hop count value will be used to factor in the switching latency and finally EIGRP’s load value will be used to determine the approximate latency likely to be caused by queuing or the queuing latency.

### 6.5.2.2 Use of EIGRP delay metric

The EIGRP delay metric represents the total delay that exists between the router and the final destination network. To calculate this value each EIGRP device along the path assigns a predefined delay metric values to all their interfaces based on the interface’s speed. Each of these calculated values along the network path is then combined in order to determine the EIGRP delay metric for the destination network. EIGRP assigns the delay value to the interface using a predefined table of values [5, 54]. For example, EIGRP will assign a 100Mbps Ethernet link a delay value of 100 microseconds. A 1000Mbps link will be assigned a 10 microsecond value and so on. The exact logic behind CISCO’s decision to use the specified predefined delay values listed could not be presented due to proprietary constraints however, the values used equate roughly to the propagation time that would be required to transmit a 1250 byte packet across the various network interface speeds with no inclusion of any small switching latency. This calculation is represented in Equation 6.3.

\[
\text{Delay Value} \approx 1250 \times \frac{8}{\text{bandwidth(bps)}} \quad (6.3)
\]

The problem with using CISCO’s predefined delay values is that when smaller packets are transmitted across the network, the predefined delay values used by EIGRP are not indicative of the actual latency smaller packets would experience. As DCCP will be used for real time data which will typically use smaller more frequent
data packets, the standard EIGRP delay values are not by default suitable. Table 6.1, which shows how packet size influences the latency in the network illustrates this.

<table>
<thead>
<tr>
<th>Link Speed</th>
<th>EIGRP Predefined Delay Value (milliseconds)</th>
<th>Expected Delay Values 180by packet (milliseconds)</th>
<th>Expected Delay Values 1500by packet (milliseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ethernet [10Mbps]</td>
<td>1000</td>
<td>144</td>
<td>1215</td>
</tr>
<tr>
<td>FastEthernet [100Mbps]</td>
<td>100</td>
<td>14.4</td>
<td>121.6</td>
</tr>
<tr>
<td>GigabitEthernet [1000Mbps]</td>
<td>10</td>
<td>1.44</td>
<td>12.16</td>
</tr>
<tr>
<td>Serial [1.544Mbps]</td>
<td>20000</td>
<td>936</td>
<td>7772</td>
</tr>
</tbody>
</table>

Table 6.1: Cisco EIGRP default delay metrics vs. actual delay metrics.

From Table 6.1 it can be seen that the default EIGRP delay values can vary significantly to the actual delay that small packets which are 180 bytes in size will experience due to the fact the EIGRP values are based roughly on a 1250 byte packet. These latency figures are based purely on interface transmission rates and do not take into consideration queuing and other delays that can occur during transmission at this stage. In order to make the reported EIGRP delay value more suitable to smaller packets the delay value is scaled in this PBS mechanism according to the packet’s size in order to provide more accurate and representative delay values. To do this the following formula is used.

\[
Delay = \frac{PacketSize}{1250} \times Default \text{ EIGRP delay value} \tag{6.4}
\]

By factoring the reported EIGRP delay value by the actual packet size as shown in the formula above, realistic wire delay latency values are obtained for the TNTCN formula.

### 6.5.2.3 EIGRP load metric

Using the delay metric alone constitutes only the wire latency portion of the total expected latency that will occur between the BOD and the destination network. As mentioned earlier, the incorporation of queues in the network path introduces an indeterminable amount of latency due to queue size fluctuations. The inability to predict how long packets will remain in queues for during transfer across the network...
makes the task of predicting their probability of arriving within their useful lifespan problematic. One approach that could be utilized to make queuing latency deterministic is to adopt a worst case scenario approach and assume queues are always full and therefore always apply the maximum latency the queue could produce if fully utilized to the probability equation formula. While this would work, this approach was not deemed feasible because it would require that all devices have knowledge of upstream device queue sizes. If this approach was taken, it may be more beneficial to simply create a separate communication channel between devices to exchange queue depth information. In addition, during times of no congestion, a packet would experience little or even no added latency due to queuing and therefore this approach would be inaccurate during non-congested periods on the network. In order to overcome this non-deterministic element in a more suitable manner, this PBS mechanism makes use of the EIGRP load metric. The EIGRP load metric is calculated dynamically by EIGRP and represents the level of saturation that exists on links along the network path to the destination network [53]. The load value is reported as a value between 1 and 255 with 1/255 representing a completely non-saturated network and 255/255 signifying a completely saturated network. In order to calculate this value, EIGRP uses a moving average of saturation levels over a five minute period with sampling occurring every five seconds [53]. This large sampling period prevents sudden saturation increases in network load from causing instability in this PBS mechanism. In addition, the frequent five second sampling rate ensures there is still accurate reflection of network conditions in the load value.

EIGRP’s load metric is ultimately indicative of congestion events occurring in the network. Wherever congestion occurs, the subsequent result is typically always an increase in queuing at various points along the network path. The larger the queue sizes become along the network path, the longer packets take to reach their intended destination. When there is a load value of 1/255, there is no congestion in the network and the time needed for a packet to reach the destination will be determined almost entirely by the wire line latency and the switching latency. As queue sizes and link saturation values begin to increase so too does the EIGRP load value. In order to estimate the time that a packet will need to traverse the various queues the EIGRP value is incorporated into the TNTCN probability equation. Through the experimentation described in previous chapters (Chapters 3-5), it was found that it is nearly
always the case that the slowest link in the network will form a bottleneck when network congestion events occur. On either side of the bottleneck, queue sizes are almost always negligible compared to the device causing the bottleneck, particularly in an upstream direction. This PBS mechanism takes this phenomenon into consideration and uses a combination of the EIGRP bandwidth metric and the EIGRP load metric to estimate the approximate queue time that a packet can expect. To calculate the queuing latency, the following formula is used.

\[
\text{Queuing Latency (seconds)} = \frac{\text{Load}}{255} \times \frac{(\text{Default queue size} \times \text{Pkt Size (Bits)})}{\text{EIGRP BW}}
\]  

(6.5)

The first step carried out by the PBS mechanism is to obtain the default queue size of the device. On Cisco devices, the default outgoing queue size is set at 40 packets [52]. The next step is to calculate the size in bits that could be in that queue. This is problematic as the packets in the queue are governed by the MTU value and cannot be known. Assuming all the packets are of the MTU size could increase the queuing delay value inaccurately if a number of small packets exist. For this reason a bias that will result in smaller than potentially possible delay values is used. The size of the current packet is used and not the MTU. If the packet is 180 bytes this PBS mechanism will assume all the packets at the bottleneck are of 180 bytes. In addition to this the queue size on the local BOD is used instead of what the actual queue size may be on the device with the lowest bandwidth. This is done because queue depth information is not exchanged in EIGRP. Once the queue size (in bits) is determined the formula then takes the slowest link in the network path obtained from EIGRP bandwidth metric and calculates the maximum time it will take a queue to be emptied based on the interface speed. The EIGRP bandwidth value represents the slowest link speed in the network path. Finally this time is weighted against the reported EIGRP load value to provide a realistic queue delay value. In the experiment section of this chapter (Section 6.7), the accuracy of the PBS mechanism is measured and it will be shown that this formula and the weighting used produced accurate estimates of the actual delay that occurs as a result of queuing.
6.5.2.4 EIGRP hop count metric

The final element of latency that must be taken into consideration is switching latency. As previously mentioned, this latency is added through the processes that take place on intermediate devices along the network path such as routing and re-encapsulation. To calculate the switching latency value, the EIGRP hop count metric is used in this PBS mechanism. The value produced by this metric represents the number of layer 3 routing devices that exist between the BOD and the destination network. The difficulty here is determining what latency each hop or layer 3 device will add to the total latency as there can be large differences in layer 3 device performance. This value can never be exactly deduced due to continual improvements in router speeds and differences in router configurations. In order to obtain an estimate, five Cisco routers were selected and inserted into a simple topology using the Opnet simulator. All settings were configured to the defaults that would be found on out of the box devices. The time it took for a packet to be transferred from one interface to another via the layer 3 routing process was then measured. Table 6.2 shows the results that were derived from this simple experiment. These measurements were based on a 1500 byte IPv4 frame.

From the values in Table 6.2 it can be seen there is variation between switching latency that will occur between the different CISCO routing devices. The probabilistic based scheduling algorithm will adopt a best case scenario approach to ensure packets are not placed into the incorrect queue due to an inaccurate result. As such each hop adds 8.2 microseconds to the TNTCN value. In practice the reality is that compared to the queuing latency component, the switching latency value is comparatively negligible. A typical switching latency is 8.2 microseconds compared to queue latency which is often 100 milliseconds and above. Even though this value is negligible it is

<table>
<thead>
<tr>
<th>Device</th>
<th>Time (microseconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>7200 Series</td>
<td>8.205</td>
</tr>
<tr>
<td>3925 Series</td>
<td>10.782</td>
</tr>
<tr>
<td>2950 Series</td>
<td>11.197</td>
</tr>
<tr>
<td>2811 Series</td>
<td>11.299</td>
</tr>
<tr>
<td>1541 Series</td>
<td>13.443</td>
</tr>
</tbody>
</table>

Table 6.2: Measured switching latency derived from simulation using a standard 1500 byte IPv4 frame.
strongly felt that it should be included in the calculation in order to provide the most accurate TNTCN value possible.

In order to provide a future proof mechanism that allows the switching time to be accurately calculated when faster switching technologies eventuate, it is recommended that each BOD should perform a quick calculation of its own switching time during initialization. One suggested implementation for doing this would be to employ two logical loopback interfaces configured on different IP subnetworks on the BOD. The protocol could then measure the time taken for packets to travel between these loopback interfaces, which would be indicative of the switching time of the BOD. This functionality is not implemented into the proof of concept model used in the experimentation. Instead the 8.2 microsecond value is used for each hop.

One disadvantage of using EIGRP’s hop count metric is that it does not take into consideration layer 2 devices such as switches. While switches will typically operated at much faster speeds than routers, there is no way in which hop count can be used to detect the existence of switches in the network path.

6.5.3 TNTCN formula

Having described where the various elements that make up the total latency value are sourced from using the EIGRP metrics, the final equation used to calculate the TNTCN variable is shown in two variations below:

\[
\text{Total delay} = (\text{Wire latency} + \text{Queuing latency} + \text{Switching latency}) \tag{6.6}
\]

OR

\[
\text{Total delay} = \left\{ \left( \frac{PS}{1250} \times RD \right) + \left( \frac{LD}{255} \times QS \times PS \right) \right\} + (HC \times 8.2M) \tag{6.7}
\]

Where:
- TD – Total delay
- PS – Packet size (Bits)
- RD – Reported EIGRP delay value
- LD – Reported EIGRP load value
- QS – Default queue size
- BW – Reported EIGRP bandwidth value
- HC – Reported EIGRP hop count
- M = Microseconds
It is important to remember that these metrics are used to merely estimate the
time a packet will need to reach its destination network. As network conditions are
continuously changing, it is unlikely this PBS mechanism will always provide perfect
results. For this reason, the PBS mechanism takes a best case scenario approach to
determining this time. This means the estimated TNTCN is biased to being slightly
ambitious because doing this gives the packets a higher probability of arriving within
their specified lifespans. Taking a worst case scenario approach which provides higher
estimated TNTCN times means that packets will have lower probabilities of being
deemed able to arrive in time and as a result a higher number of packets will be
placed in the discard or critical queue unnecessarily. Being too lenient and providing
too much leeway in the calculation will mean that the PBS mechanism loses its
effectiveness. The experimentation in Section 6.7 will show that the weighting and
selection of the various metrics presented above provides a TNTCN value that is
sufficiently accurate for use in the PBS mechanism.

6.5.3.1 Alternative approaches considered for estimating TNTCN

Another approach that was considered as an alternative method for determining
the TNTCN value was to make use of DCCP’s RTT value. There are however some
major problems with this approach which will now be discussed. The first issue is
that utilizing RTT values would require all the BOD devices to maintain RTT tables
for every flow that traverses the device. This would be computationally intensive and
not scalable in very large networks. In addition to this the RTT value is prone to high
levels of fluctuation and basing the predicted TNTCN value on a single RTT would
lead to a very high level of incorrectly categorized packets. Instead a moving average
of RTT values would be required in order to ensure the PBS mechanism did not
become reaction to single short lived changes in the network. The need to maintain
a moving average of RTT values for every individual flow traversing the network was
again deemed too computationally intensive given the inaccuracy of the RTT value in
general. Finally such an approach would leave the BOD device vulnerable to denial
of service type attacks where malicious devices opened larger numbers of connections
in order to over utilize memory on the BOD. Doing this could render the network
6.6 Servicing queues

Once the categorization and ranking tasks described above have been performed, the next step needed is to forward packets from the three queues on to their intended destinations. While this mechanism does allow flexibility in the way the queues can be serviced, the following rules must be adhered to. The critical queue must be allocated a predefined percentage of bandwidth on the outgoing interface prior to the commencement of any transfers. While there are packets in the standard queue, the critical queue is never to exceed this threshold. In the unlikely event that the standard queue becomes empty, some critical queue packets may be placed in the normal queue. These must not however cause any detriment to the deterministic nature of the normal queue.

In addition to this rule, packets placed in the normal queue are never to be re-ordered. This queue is to be configured as a FIFO queue due to the construction of the categorization algorithm. Changing the order of packets once in the queue will greatly increase computational overhead and will jeopardize the deterministic nature of the queue which provides its strength. The critical queue and the discard queue may use sophisticated re-ordering algorithms to order packets in their respective queue. Packets in the discard queue are to be given the lowest priority in times of heavy congestion and are to be dropped first if mechanisms such as RED are used on the devices. As shown in Chapter 5 Section 5.4.1.2, simply discarding all stale packets may lead to unstable DCCP operation and therefore a mechanism such as the hybrid discard purging mechanism should be utilized in the discard queue. In order to facilitate such a mechanism, a certain amount of bandwidth would be required to allow ECN marked packets to traverse the network.

If the destination network does not exist in the routing table then the packet is to be placed in the normal queue. In addition, if no lifetime information is present in the packet or if the packet is marked with a non-time sensitive option value of 99999, then the packet is to be placed into the normal queue. Finally if the packet is not

unavailable to all non-miscreant traffic flows. For these reasons, it was decided the approach whereby information was gleaned from the routing protocol was more robust and accurate and hence the decision to use that approach was taken.
utilizing DCCP at the transport layer (i.e. TCP/UDP), it is also to be placed in the normal queue.

6.6.1 Theoretical example

To clarify the theory presented in the preceding sections, a theoretical example will now be given. Before the commencement of data transfer the three queues are established on all outgoing network interfaces on the BOD. It is assumed that the BOD is aware of the link speed and available bandwidth on each of the outgoing interfaces. Once this information is known, the BOD is to allocate maximum predefined bandwidth limits to each of the queues using either a bucket or token based scheduling mechanism. For example, the BOD could assign 25% of the available bandwidth to the critical queue, 70% of the available bandwidth to the normal queue and the final 5% to the discard queue. The actual allocation will take place in the same fashion as CISCO allocates LLQ queue proportions where the administrator configures the desired values via the devices command prompt.

By setting predefined hard limits on the bandwidth allocated to each queue, the maximum time needed for a packet to be in the queue can be calculated (if the order of the packets in the queue does not change). Each outgoing interface will exist as a separate and independent entity and as such each interface will have its own critical queue, a normal queue and a discard queue. After successful transmission of a packet, the scheduler mechanism will access and service packets from each of these queues according to the bandwidth allocation settings. The way in which packets in queues are serviced will now be discussed.

6.6.2 Scheduling implementation on the BOD

As mentioned previously, any scheduling mechanism could be used in conjunction with this PBS mechanism to schedule packet delivery provided it follows the guidelines above. In this section, the queue scheduling mechanism selected for this proof of concept modeling will be discussed. For the purposes of implementing the PBS mechanism, statistical time division multiplexing [55] was selected as the method for scheduling packet delivery on outgoing interfaces. With statistical multiplexing, each
of the queues essentially became a channel in the scheduling algorithm. Each channel was then assigned the predetermined amount of bandwidth and serviced at a fixed rate accordingly. The advantage of statistical multiplexing over traditional time division multiplexing [56] is that where a particular channel does not need to transmit during its pre-allocated slot, this slot can then be used by the next channel waiting to transmit.

In traditional time division multiplexing, this does not occur and the slot remains unused rather than being allocated to the next channel. For the purposes of the experimentation, the Opnet Modeler simulation toolkit provided statistical time division multiplexing modules as well as the queuing modules that were added and configured to service the queues on outgoing interfaces on the BOD device. For the purposes of this PBS mechanism, each of the three queues was configured as an individual channel and serviced based on a strict bandwidth allocation percentage. Based on this bandwidth percentage allocation, each channel was then issued slots in which packets residing in the respective channel queue could then be serviced. As mentioned previously, if a particular channel did not possess any packets that required servicing then the slot was allocated to the next channel due to be allocated a slot.

6.7 Experimentation: Examining the effectiveness of the PBS mechanism

This section describes four separate experiments that were carried out to examine and test the PBS mechanism described in this chapter. In the experiments described in previous chapters, the experimental simulated topology model was intentionally designed to be simplistic in order to gather results that were used to analyze very specific phenomena such as contention and fairness in the network. In the experimentation described in this chapter, the complexity of the simulated topology was increased because of the introduction of EIGRP. In particular, the simulated topology in this experimentation contained more intermediate devices, such as routers, that were connected through various speed links. In addition to this, each of the experimental topologies also had multiple simultaneous flows passing through the network at any given time, in the order of 10, 20, 50 and 100 simultaneous flows. This increase
in complexity was necessary due to the inclusion of EIGRP into the PBS mechanism and the need to have networks that were complex enough to provide larger routing tables and traffic loads that were capable of creating congestion events.

While the complexity increased the realism of the simulated environment, it also meant it became increasingly more difficult to attribute specific results or findings purely to the function of the PBS mechanism. Due to the complexity of DCCP and CCID3, it would not be valid to simply compare results from normal DCCP operation with the results obtained using the PBS mechanism and attribute any differences in these to the function of the PBS mechanism. For example, if one individual flow in the topology performed better where the PBS mechanism was used compared to the same flow where the PBS mechanism was not utilized, it could not be said that the PBS mechanism was purely responsible for that improvement. The actual cause may have been due to a change in contention dynamics between the various flows or because another flow had been severely limited or affected by the PBS mechanism freeing up bandwidth for the flow being measured. It would not be valid to make such simplistic inferences because of the complexity of the topology and complex interrelationships that existed between the multiple flows.

In addition to this, through the experience of running multiple simulations on the various simulated topologies, it quickly became apparent that the effectiveness, or lack thereof, of the PBS mechanism was heavily dependent on the configuration of the topology. If it were possible to measure the benefit of the PBS mechanism, the results would only be valid for that particular set of network conditions. Instead of trying to derive a single quantitative value to measure the PBS mechanism’s potential improvement yield, the experimentation examined 20 simulated topologies and sought to determine what conditions the mechanism appeared to operate optimally in, and in what conditions the mechanism offered no benefit or reduced DCCP performance.

6.8 Experiment 1: Accuracy of probability TNTCN calculation

In this experiment the accuracy of the formula used to derive the TNTCN value was explored. To achieve this, the experimentation compared the computed TNTCN values to the actual time it took for the packets to reach the destination network. This
comparison was used to determine if the weighting used in the TNTCN formula was suitable for the purposes of the PBS mechanism by ensuring the TNTCN values and actual values were within acceptable range. In addition to this, if there was variance in the times, the TNTCN formula should have provided a bias towards generating lower TNTCN values than the actual time it took for a packet to cross the network. If the TNTCN formula produced excessively large TNTCN values then packets could inadvertently be placed into the critical queue or even worse be deemed unable to reach the destination within the remaining lifespan and placed in the discard queue. By maintaining a bias towards generating smaller TNTCN values, the PBS mechanism would produce fewer incorrectly categorized packets. In order to meet the design criteria, the TNTCN and actual times it took packets to cross the network needed to be within a 10% threshold of difference during normal network operation periods. This experiment tested to ensure this was the case.

6.8.1 Experiment methodology

As mentioned previously, 20 simulated topologies were created for this experimentation with each topology possessing varying levels of complexity. In all of these topologies there were at least five simultaneous flows passing through the network from various end points (servers). In order to gather the data needed for this analysis, five randomly selected flows were chosen from each simulation where more than five flows were present on the topology. The experiments were run over a simulated period of five minutes. The data for the experiments were only collected during the final 60 seconds of the five minute simulated period. This was done to allow sufficient time for the routing protocol to converge and network load or congestion levels to reach stabilized levels.

Using global variables in the Opnet Modeler simulator, every time a packet left the BOD, a timestamp entry was created in a predefined global variable array. In addition to this, the TNTCN value calculated by the PBS mechanism was also stored in the same global variable array. Once the packet arrived at the destination network, its sequence number was used to access the corresponding global variable and the current NTP time was entered into the variable store. An example of this statistical global variable array structure and elements contained within it is shown in Figure 6.2.
Figure 6.2: Array structure used to store global variables needed to test TNTCN accuracy.

The last routing devices along the network path were configured to add the simulation_time_pkt_arrive value to the global array when a packet arrived at the incoming interface on the router. The reason for configuring the last router to perform this task, and not the destination workstation, was because the EIGRP metric calculates the time for the packet to reach the destination network only and not the actual endpoint device. At the end of the simulation period all the global variable arrays were collected. From these variable arrays the actual time the packet took to traverse the network to the final destination network was calculated (simulator_pkt_depart – simulator_pkt_arrive). The actual time that had been taken and the calculated time needed (TNTCN) were then compared. In a few instances some of the packets were discarded by an upstream BOD device where the packet was deemed unable to traverse the network within its useful lifetime or where the packet had become stale. In order to accommodate this phenomenon in the results, wherever the Simulator_time_pkt_arrive value was not recorded, the entire global variable array relating to that particular packet was discarded. All measurements were taken from BOD 1 in instances where more than one BOD device resided in the topology.

Results from the 20 simulated topologies were collected and five flows from each of these topologies were monitored specifically. This has led to a collection of results for 100 separate flows. In the interests of keeping this chapter concise not all 100 flows will be discussed in great detail. Instead, a summarized version of the results is presented in Table 6.3.

6.8.2 Results

In Table 6.3, the percentage of average variation found between the TNTCN calculation and the actual time it took packets to reach the destination network is presented.
To calculate the values shown, the percentage of difference was first calculated for each packet variable array \((TNTCN - \text{Actual time} \times \frac{100}{1})\). This value represents the percentage difference in times between the calculated TNTCN value and the actual time that was measured. Once this had been done for all the entries, the average of these values was calculated. To do this, all the percentage of differences calculated in the previous step were summed and then averaged \((\text{Sum(percentage of difference)} / \text{number of calculations})\). Note a “+” symbol indicates that the average TNTCN value was larger than the actual time taken value and a “−” symbol indicates the TNTCN value was found to be smaller on average than the actual time taken.

In topologies 14 and 17 (shown in Table 6.3), a T3 serial link was placed into the network topology, which resulted in an issue with the PBS mechanism being discovered. The delay value automatically assigned to all serial links in an EIGRP topology is 20000 microseconds by default, irrespective of the speed of the serial link. When this value was used in the TNTCN formula, there was a large discrepancy between the calculated TNTCN value and the actual time it took the packet to reach the destination network. For the remainder of this narrative, results from topologies 14 and 17 will be excluded because of this anomaly.

As can be seen from the results in Table 6.3, in all flows the calculated TNTCN

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-3.210%</td>
<td>-2.922%</td>
<td>-3.749%</td>
<td>-4.542%</td>
<td>-2.8924%</td>
</tr>
<tr>
<td>2</td>
<td>-3.433%</td>
<td>-3.879%</td>
<td>-3.773%</td>
<td>-4.185%</td>
<td>-3.860%</td>
</tr>
<tr>
<td>3</td>
<td>-4.371%</td>
<td>-3.823%</td>
<td>-4.067%</td>
<td>-4.592%</td>
<td>-3.376%</td>
</tr>
<tr>
<td>4</td>
<td>-3.483%</td>
<td>-3.653%</td>
<td>-3.833%</td>
<td>-4.042%</td>
<td>-3.325%</td>
</tr>
<tr>
<td>5</td>
<td>-6.084%</td>
<td>-6.399%</td>
<td>-6.545%</td>
<td>-6.423%</td>
<td>-6.284%</td>
</tr>
<tr>
<td>6</td>
<td>-1.218%</td>
<td>-0.924%</td>
<td>-1.188%</td>
<td>-1.255%</td>
<td>-0.827%</td>
</tr>
<tr>
<td>7</td>
<td>-0.313%</td>
<td>-0.267%</td>
<td>-0.231%</td>
<td>-0.109%</td>
<td>-0.119%</td>
</tr>
<tr>
<td>8</td>
<td>-4.520%</td>
<td>-5.643%</td>
<td>-5.012%</td>
<td>-5.221%</td>
<td>-4.845%</td>
</tr>
<tr>
<td>9</td>
<td>-6.767%</td>
<td>-6.289%</td>
<td>-6.133%</td>
<td>-6.934%</td>
<td>-6.459%</td>
</tr>
<tr>
<td>10</td>
<td>-6.001%</td>
<td>-5.847%</td>
<td>-6.210%</td>
<td>-6.387%</td>
<td>-6.113%</td>
</tr>
<tr>
<td>11</td>
<td>-2.329%</td>
<td>-2.367%</td>
<td>-2.753%</td>
<td>-2.958%</td>
<td>-2.164%</td>
</tr>
<tr>
<td>12</td>
<td>-3.562%</td>
<td>-3.324%</td>
<td>-3.491%</td>
<td>-3.762%</td>
<td>-3.445%</td>
</tr>
<tr>
<td>13</td>
<td>-3.011%</td>
<td>-2.872%</td>
<td>-2.992%</td>
<td>-3.019%</td>
<td>-2.691%</td>
</tr>
<tr>
<td>14</td>
<td>+17.601%</td>
<td>+17.938%</td>
<td>+17.614%</td>
<td>+18.478%</td>
<td>+17.737%</td>
</tr>
<tr>
<td>15</td>
<td>-7.968%</td>
<td>-7.758%</td>
<td>-7.387%</td>
<td>-8.044%</td>
<td>-7.891%</td>
</tr>
<tr>
<td>16</td>
<td>-8.021%</td>
<td>-8.762%</td>
<td>-8.539%</td>
<td>-8.893%</td>
<td>-8.142%</td>
</tr>
<tr>
<td>17</td>
<td>+12.116%</td>
<td>+12.529%</td>
<td>+12.418%</td>
<td>+12.933%</td>
<td>+12.674%</td>
</tr>
<tr>
<td>18</td>
<td>-2.985%</td>
<td>-2.754%</td>
<td>-2.531%</td>
<td>-2.997%</td>
<td>-2.655%</td>
</tr>
<tr>
<td>19</td>
<td>-3.749%</td>
<td>-3.564%</td>
<td>-3.285%</td>
<td>-3.857%</td>
<td>-3.651%</td>
</tr>
<tr>
<td>20</td>
<td>-0.871%</td>
<td>-0.654%</td>
<td>-0.213%</td>
<td>+0.097%</td>
<td>-0.054%</td>
</tr>
</tbody>
</table>

Table 6.3: Average percentage difference between calculated TNTCN and actual time recorded.
produced on average a TNTCN value that was within the desired 10% threshold. In addition, in all but two of the flows, namely Topology 7 Flow 4 and Topology 20 Flow 4, the calculated TNTCN value had a distinct bias towards generating lower TNTCN values than the actual transmit times that occurred. Delving deeper into the results, there was clear evidence that in more complex topologies such as in topologies 7 and 20, the mechanism did at times begin to lose its bias towards generating smaller TNTCN values. When exploring the results from these topologies, it became apparent that where EIGRP load metric values increased above 200/255, the PBS mechanism began generating higher volumes of TNTCN values that were higher than the actual time it took for packets to cross the network.

From this it was concluded that where the EIGRP load value was below 200/255 the TNTCN formula had a clear bias towards generating smaller TNTCN values than the time packets took to cross the network and only very rarely generated higher values. However, once the EIGRP load value exceeded 200, the TNTCN values showed no clear bias towards being lower than the actual time it took for packets to arrive at the destination network. From Topologies 7 and 20, it can be seen that there was even a slight bias towards generating higher TNTCN values when compared to the actual time it took for the packets to reach the destination.

The next grouping of results took the results analysis one step further to determine the percentage of times that the TNTC values were higher than the actual times that were taken for the packets to arrive at the destination network. Specifically, this value shows the percentage of packets in the particular flow that are given higher than necessary TNTCN values. The danger with having a higher TNTCN value is that packets may be categorized into the critical or discard queue incorrectly. The reality for these packets is that they arrived in a faster time than the TNTCN value has predicted. Using the same results collected in the first part of the experiment, all instances where the TNTCN value was larger than the actual time that was taken for the packet to reach the destination network were identified. The number of instances where this was found to be the case were then represented as a percentage of the total number of packets that were sent during the collection period (i.e. \( \frac{\text{no. of young packets}}{\text{total number of packets}} \)). These figures are presented in Table 6.4.

In all but three topologies, namely Topology 6, 7 and 20, as shown in Table 6.4, the percentage of packets that were incorrectly categorized by the TNTCN formula
Table 6.4: Percentage of packets assigned a TNTCN value higher than the actual time taken.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
</tr>
<tr>
<td>2</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
</tr>
<tr>
<td>3</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
</tr>
<tr>
<td>4</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
</tr>
<tr>
<td>5</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
</tr>
<tr>
<td>6</td>
<td>0.001013%</td>
<td>0.002112%</td>
<td>0.00129%</td>
<td>0.00135%</td>
<td>0.00285%</td>
</tr>
<tr>
<td>7</td>
<td>0.000528%</td>
<td>0.005001%</td>
<td>0.00609%</td>
<td>0.00724%</td>
<td>0.00585%</td>
</tr>
<tr>
<td>8</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
</tr>
<tr>
<td>9</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
</tr>
<tr>
<td>10</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
</tr>
<tr>
<td>11</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
</tr>
<tr>
<td>12</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
</tr>
<tr>
<td>13</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
</tr>
<tr>
<td>14</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
</tr>
<tr>
<td>15</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
</tr>
<tr>
<td>16</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
</tr>
<tr>
<td>17</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
</tr>
<tr>
<td>18</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
</tr>
<tr>
<td>19</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
<td>&lt;0.001%</td>
</tr>
<tr>
<td>20</td>
<td>0.00341%</td>
<td>0.00244%</td>
<td>0.00407%</td>
<td>0.00553%</td>
<td>0.00285%</td>
</tr>
</tbody>
</table>

was below 0.001% of the total number of packets sent. In Topology 6, the TNTCN formula calculated a higher TNTCN value than that was actually experienced on approximately 0.002% of the total packets sent. In Topology 7 and 20, this value was approximately 0.006% and 0.004% respectively of the total number of packets sent. As can be seen, the number of times the TNTCN formula generated higher values than those that were recorded in the simulation was extremely low.

6.8.3 Analysis and discussion of results

The results show that the TNTCN formula used in the PBS mechanism, in the vast majority of cases (>90%), was able to provide values that fell within the specified threshold. In addition to this, the formula also provided a bias towards generating lower TNTCN values than were actually experienced in all but two of the 90 streams tested. This shows the given formula provides suitable TNTCN values needed by the PBS mechanism.

One of the difficulties discovered when designing the TNTCN formula was that the PBS mechanism provides merely a prediction of the expected time that is needed for the packet to arrive at the destination network and will never be able to produce
a perfectly accurate value. The results show that in the majority of cases (70%), the variance between the TNTCN value and the actual time value was on average less than 5%. In fewer cases (20%), the variance between the TNTCN and actual times taken was between 5% and 10%.

The formula used to calculate the TNTCN value could potentially be improved to provide more accurate TNTCN values. One way this could be achieved would be to incorporate an additional 6th metric into EIGRP whereby queue depths are advertised among EIGRP devices. In doing this, the need to try to estimate queuing times would be eliminated from the formula calculation which would make this mechanism more accurate. While the value could potentially be improved, the results from these experiments categorically show that for the purposes of the PBS mechanism and this proof of concept experimentation, the current formula is adequate.

The final issue discovered through this experiment was that EIGRP assigns a delay value of 20000 microseconds to all serial links in the topology when calculating the delay metric. Using this value created a high TNTCN value and led to a high number of incorrectly categorized packets. While this is problematic, the reality is such that if high speed serial links are employed on networks, typically the delay value in EIGRP is configured manually to ensure EIGRP is given an accurate representation of actual network conditions. In such cases, this PBS mechanism would simply make use of the manually configured EIGRP delay value and this issue would be resolved.

6.9 Experiment 2: Identifying optimal network conditions for the mechanism

This second experiment investigated the conditions in which the PBS mechanism provided higher levels of improvement to overall network performance when the various result topologies were examined. This experiment did not seek to provide a single numerical value stating the potential increase possible through the mechanism (i.e. such as a claim that the PBS mechanism provided a 10% increase in overall performance). This is because given the complexities of the network used in this experimentation and the immeasurable nuances caused through multiple streams simultaneously passing
through the network, this value would not be valid. Instead, the goal of this experiment was to determine the network conditions and circumstances where the PBS mechanism appeared to offer greater levels of performance compared to conventional DCCP.

Specifically, the experiments tested if given the correct network conditions, whether the PBS mechanism would provide more efficient transportation of DCCP traffic when compared to a conventional mechanism where no categorization of packets according to their respective age took place. In particular, the experiment explored if the PBS mechanism provided most benefit in situations where high levels of categorization of packets occurred and where a high volume of these packets were sent to either the critical queue or the discard queue. Finally, the experiment explored if situations where the majority of packets were placed into the normal queue, resulted in comparable performance to topologies where no categorization of packets occurred.

6.9.1 Experiment methodology

This experimentation utilized the same 20 topologies used in the previous experiment (Section 6.8). In these 20 topologies an intentional bottleneck was created along the network path which serviced all flows (see Appendix A). To create this bottleneck, a high speed link was connected to the BOD on the incoming interface and a slower link was connected to the same BOD on the outgoing interface. The topology was designed in such a way that all the server's data traveled through the incoming interface of the BOD (left of the BOD device (logically)) and out the outgoing interface to the various workstations (clients) which were placed to the right of the BOD (logically). Placing the servers in such a way was done to generate higher queue, link load and congestion levels on the BOD device. In addition to this, only one flow direction was used for the simulated traffic and the feedback flow in the opposite direction was not explored. If the servers were placed on either side of the BOD then feedback from the workstations such as acknowledgments would be delayed or impacted by the PBS mechanism which would cause inadvertent reduction in DCCP congestion window sizes or even session timeouts if acknowledgments were discarded. The BOD devices were configured to only utilize the PBS mechanism on flows traveling from servers to
the workstation. The return flows were not impacted and standard FIFO queuing was applied to packets traveling in the return direction. Figure 6.3 illustrates the logical topology representing the flow topology design.

Having created the bottleneck in the topology, the way in which the categorization and servicing of the congested queues took place was examined to determine if there were any noticeable performance gains offered by the PBS mechanism. Where such improvement was found, analysis then sought to determine if there were any specific factors or criteria that led to these increases.

In addition to this, scenarios where the mechanism performed less optimally and in similar fashion to standard DCCP were also examined to see if there were factors that led to lower performance gains as a result of the actions of the PBS mechanism. To analyze these factors the queue levels and the packets contained within the queues on BOD devices on the topology were monitored for 10 minutes after the initial first minute of transfers and simulation initialization. After the first minute had elapsed, collection of results began. This was done to allow DCCP send rates to stabilize as well as allowing the EIGRP routing protocol to converge. The results collected during this 10 minute phase were then compiled at the end of the simulation.

In addition to collecting results relating to the various queues on the BOD devices, the number of packets received beyond their advertised lifespan was also collected from all workstations in the topology. This was done using a collection bucket and a single global variable in the simulator. The workstations in the topology were configured to add one to the current value of the total_stale_pkts_rcvd global variable whenever a stale packet was received. At the end of the simulation, the total number of stale packets received was collated. These values were then used to determine if the PBS mechanism reduced or increased the number of packets that were received beyond their useful lifespan. The final statistic that was collected from the experiment was the
average age of the packets received at the workstations. Each of the workstations were configured to maintain the average age of all packets received during the simulation. At the end of the simulation, these values were also collected from all the workstations and analyzed.

6.9.2 Results

6.9.2.1 Queue depth levels

The queue depth level results show the number of packets that were serviced by each of the three queues during the simulation. From these results the topologies that lead to higher volumes of traffic entering into particular queues were identified. In the analysis section of this experiment (Section 6.9.3), the factors that lead to these characteristics will be discussed. Table 6.5 shows a summary of the queue depths that were recorded from the experiments.

From the results shown in Table 6.5, it can be seen in all the 20 tested topologies, the normal queue serviced the majority of all packets during the simulation period (between 77.53\% - 87.01\% of all packets). The critical queue serviced the second highest number of packets in all of the tested topologies (between 8.46\% - 16.78\% of all packets), while the discard queue serviced the least number of packets in all of the simulated topologies (between 4.53\% - 8.45\% of all packets).

The next step taken was to identify the topologies where the highest recorded percentages of packets that were placed into the various queues were found. By doing this, the characteristics found in each of those topologies could then be explored to determine the configurations that contributed to the higher queue utilization levels.

The highest percentage of packets placed into the normal queue compared to the other topologies were experienced in Topologies 5, 10 and 20. In Topologies 14, 15 and 16 the highest percentages of packets placed into the critical queue compared to the other topologies were experienced. In Topologies 3, 14 and 16 the highest percentage of packets placed into the discard queue compared to the other topologies were recorded.

Having identified the topologies where each of the queues received high percentages of the total packets for the flows, the next step taken was to determine what in these
<table>
<thead>
<tr>
<th>Topology</th>
<th>Discard Queue</th>
<th>Normal Queue</th>
<th>Critical Queue</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>3624 (5.81%)</td>
<td>50742 (81.33%)</td>
<td>8022 (12.86%)</td>
</tr>
<tr>
<td>2</td>
<td>3215 (6.27%)</td>
<td>42025 (81.89%)</td>
<td>6080 (11.85%)</td>
</tr>
<tr>
<td>3</td>
<td>2191 (8.39%)</td>
<td>20679 (79.15%)</td>
<td>3257 (12.47%)</td>
</tr>
<tr>
<td>4</td>
<td>6552 (5.90%)</td>
<td>91723 (82.61%)</td>
<td>12758 (11.49%)</td>
</tr>
<tr>
<td>5</td>
<td>8569 (6.06%)</td>
<td>119972 (84.85%)</td>
<td>12847 (9.09%)</td>
</tr>
<tr>
<td>6</td>
<td>1991 (5.89%)</td>
<td>27867 (82.42%)</td>
<td>3953 (11.69%)</td>
</tr>
<tr>
<td>7</td>
<td>1856 (6.03%)</td>
<td>25980 (84.89%)</td>
<td>2779 (9.08%)</td>
</tr>
<tr>
<td>8</td>
<td>8734 (5.87%)</td>
<td>122269 (82.24%)</td>
<td>17668 (11.88%)</td>
</tr>
<tr>
<td>9</td>
<td>9393 (5.95%)</td>
<td>131505 (83.30%)</td>
<td>16966 (10.75%)</td>
</tr>
<tr>
<td>10</td>
<td>8497 (6.07%)</td>
<td>118959 (85.01%)</td>
<td>12485 (8.92%)</td>
</tr>
<tr>
<td>11</td>
<td>3417 (5.86%)</td>
<td>47841 (82.00%)</td>
<td>7086 (12.15%)</td>
</tr>
<tr>
<td>12</td>
<td>3711 (5.32%)</td>
<td>51954 (82.98%)</td>
<td>8152 (11.69%)</td>
</tr>
<tr>
<td>13</td>
<td>3903 (5.82%)</td>
<td>54644 (81.48%)</td>
<td>8516 (12.70%)</td>
</tr>
<tr>
<td>14</td>
<td>2951 (7.17%)</td>
<td>31307 (76.05%)</td>
<td>6910 (16.78%)</td>
</tr>
<tr>
<td>15</td>
<td>4439 (5.71%)</td>
<td>62145 (79.95%)</td>
<td>11150 (14.34%)</td>
</tr>
<tr>
<td>16</td>
<td>10359 (8.45%)</td>
<td>95026 (77.53%)</td>
<td>17180 (14.02%)</td>
</tr>
<tr>
<td>17</td>
<td>3512 (5.83%)</td>
<td>49175 (81.62%)</td>
<td>7562 (12.55%)</td>
</tr>
<tr>
<td>18</td>
<td>6823 (5.66%)</td>
<td>99528 (82.62%)</td>
<td>14117 (11.72%)</td>
</tr>
<tr>
<td>19</td>
<td>9488 (5.94%)</td>
<td>132837 (83.14%)</td>
<td>17442 (10.92%)</td>
</tr>
<tr>
<td>20</td>
<td>1916 (4.53%)</td>
<td>26820 (87.01%)</td>
<td>3579 (8.46%)</td>
</tr>
</tbody>
</table>

Table 6.5: Queue depths recorded from simulation.
topologies led to the higher utilization of the respective queues. In Topologies 14 and 16, there was a large delay experienced between the sender and the BOD device. As such, when packets arrived at the BOD, they had already been exposed to a large amount of delay due to the network conditions. This meant the packets were nearing their maximum TTL value before any action could take place on them. As a result of their age, a comparatively high volume of packets were either placed into the discard queue or the critical queue as is shown in Table 6.5. For this reason, both the discard queue and the critical queue had higher percentages of packets in the flow when compared to the other topologies.

In Topology 3, one of the flows measured was subjected to a large delay prior to the BOD due to the network configuration between the sender and the BOD. The other four flows measured in the topology were not subjected to this same delay. The results which show an unusually high number of stale packets are due to the flow which was subjected to the large delay delivering packets that were very near to the 200ms age limit to the BOD. The network delay experienced by the flow prior to the BOD was measured to be 185ms. As a result the BOD placed the majority of the packets arriving from this flow into the discard queue.

In Topology 15, it was again observed that the delay values caused by network configurations led to a high number of packets being placed into the critical queue. In Topology 15, the delay between the senders and the BOD was between 150ms and 175ms for each of the five flows utilizing the network. As most of the packets could still be delivered within their useful lifespan they were placed into the critical queue for prioritized processing.

One of the most interesting phenomenon noted from the results appears in Topologies 5, 10 and 20. In these topologies the network conditions are such that the normal queue is utilized at a much higher percentage level when compared to the other topologies. The reason for this is due to the fact that in Topologies 5, 10 and 20 there was little delay caused by the network prior to the BOD. As such, most of the packets were categorized into the normal queue as they were not in danger of expiring. What is of interest in this scenario is the underutilization of both the discard queue and critical queue when this occurs. When the critical queue and the discard queue are not used, the ability for the PBS mechanism to improve DCCP performance when compared to normal DCCP operation is reduced. As such, in topologies where pack-
Table 6.6: Average number of stale packets per workstation (per 1000 Packets).

<table>
<thead>
<tr>
<th>Topology</th>
<th>Average Number of stale packets per workstation (per 1000 packets)</th>
<th>Number of workstations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>4.86</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>3.42</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>5.23</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>3.49</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>4.17</td>
<td>25</td>
</tr>
<tr>
<td>6</td>
<td>8.54</td>
<td>20</td>
</tr>
<tr>
<td>7</td>
<td>10.88</td>
<td>20</td>
</tr>
<tr>
<td>8</td>
<td>1.55</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>0.97</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>3.81</td>
<td>20</td>
</tr>
<tr>
<td>11</td>
<td>3.61</td>
<td>20</td>
</tr>
<tr>
<td>12</td>
<td>5.19</td>
<td>20</td>
</tr>
<tr>
<td>13</td>
<td>4.87</td>
<td>20</td>
</tr>
<tr>
<td>14</td>
<td>5.32</td>
<td>25</td>
</tr>
<tr>
<td>15</td>
<td>3.63</td>
<td>25</td>
</tr>
<tr>
<td>16</td>
<td>3.47</td>
<td>25</td>
</tr>
<tr>
<td>17</td>
<td>8.82</td>
<td>5</td>
</tr>
<tr>
<td>18</td>
<td>7.64</td>
<td>5</td>
</tr>
<tr>
<td>19</td>
<td>7.98</td>
<td>5</td>
</tr>
<tr>
<td>20</td>
<td>11.21</td>
<td>20</td>
</tr>
</tbody>
</table>

From the results it can be seen that the average number of stale packets received at a young age, based on the results in Table 6.5, the benefit of the categorization mechanism was found to be reduced to levels providing similar performance to standard DCCP. The analysis section in Section 6.9.3 will expand on this finding.

6.9.2.2 Number of stale packets received

This section presents results that show the average number of stale packets that were received by the endpoint workstation devices in each of the topologies. Because there were varying numbers of workstation devices in each of the topologies, the average number of stale packets received by the workstations is presented in the results shown in Table 6.6. In order to calculate these values, the total number of stale packets received was divided by the number of workstations participating in the topology. This value was then adapted to show how many packets were found to be stale on each workstation for every 1000 packets received.

From the results it can be seen that the average number of stale packets received...
by each workstation varied between 0.97 packets and 11.21 stale packets per 1000 packets. The reasons for these values will be presented in Section 6.9.3.

6.9.2.3 Average packet age

This section presents results that show the average packet age of packets when they are received by the endpoint workstation devices. For the purposes of this experimentation, 200ms was configured to be the maximum lifespan value for each of the data packets. As was the case in the results section in Section 6.9.2.2, because of the difference in number of workstations found in each of the topologies, the average age of packets from all the workstations in the topology was calculated. In Table 6.7, the average age of packets received by the endpoint workstations is shown.

The average age of packets received by the workstations was found to vary between 43ms and 189ms. From this, it can be seen that the average age of packets received by the workstation was within the required 200ms threshold. It is important to note

<table>
<thead>
<tr>
<th>Topology</th>
<th>Average packet age (ms)</th>
<th>No. of workstations</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>109</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>148</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>170</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>87</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>56</td>
<td>25</td>
</tr>
<tr>
<td>6</td>
<td>184</td>
<td>20</td>
</tr>
<tr>
<td>7</td>
<td>189</td>
<td>20</td>
</tr>
<tr>
<td>8</td>
<td>50</td>
<td>10</td>
</tr>
<tr>
<td>9</td>
<td>43</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>68</td>
<td>20</td>
</tr>
<tr>
<td>11</td>
<td>128</td>
<td>20</td>
</tr>
<tr>
<td>12</td>
<td>103</td>
<td>20</td>
</tr>
<tr>
<td>13</td>
<td>119</td>
<td>20</td>
</tr>
<tr>
<td>14</td>
<td>162</td>
<td>25</td>
</tr>
<tr>
<td>15</td>
<td>95</td>
<td>25</td>
</tr>
<tr>
<td>16</td>
<td>48</td>
<td>25</td>
</tr>
<tr>
<td>17</td>
<td>133</td>
<td>5</td>
</tr>
<tr>
<td>18</td>
<td>76</td>
<td>5</td>
</tr>
<tr>
<td>19</td>
<td>47</td>
<td>5</td>
</tr>
<tr>
<td>20</td>
<td>179</td>
<td>20</td>
</tr>
</tbody>
</table>

Table 6.7: Average age of packets.
that packets received on the BOD that were stale were purged and therefore such packet statistics are not accounted for by these results.

6.9.3 Analysis and discussion of results

The results gathered from this experimentation show that where the topology was conducive to generating higher volumes of packets that were closer to their maximum age, the PBS mechanism offered the greatest improvement in DCCP performance. In contrast with this, the results also indicated that there were moderate to no gains in performance provided by the mechanism, where the majority of packets were categorized into the normal queue. There was however, no noticeable detriment experienced between the PBS mechanism and normal DCCP when all packets were sent to the normal queue. Where there were a proportionally high number of packets placed in the discard queue, there were some stability issues caused by assigning too little bandwidth to the critical queue initially that were not shown in the experimentation but were discovered during the implementation of the scheduler. If the discard queue was made too small and a high number of packets were dropped to prevent queue overflows, the instability issues described in Chapter 4 reappeared. At approximately 5% of the total bandwidth however, which was used in all the above experimentation, none of the streams timed out as a result of an excessive number of packet drops.

6.10 Experiment 3: Scalability of the mechanism

The third experiment presented in this chapter explored both the PBS mechanism's scalability and its ability to remain stable for extended durations. To do this, multiple BOD devices were placed into a three large simulated network topologies and the simulations were run over an extended period. This tested the PBS mechanism's ability to handle a range of network conditions as well as the ability for multiple BOD devices to be placed into a single network path. The PBS mechanism was designed specifically to ensure that multiple BOD devices could coexist on a single network path and that no conflict would result from such scenarios. Additionally, this
experiment was carried out to ensure that the actions carried out by the BOD allowed for long lived DCCP transfers. Finally, this experiment measured the computational overheads that were needed for the PBS mechanism. To do this, CPU and memory were monitored on the BOD to determine if the PBS mechanism was computationally feasible on currently available network hardware.

6.10.1 Experiment methodology

In this experiment three complex network topologies were used to carry out the experimentation. Each of the topologies were used to carry out simulations that were configured to run for a period of 12 simulated hours during which the levels of congestion occurring in the network were made to fluctuate from high to low. Using moderately slow links between the various subnetworks, the topologies were deliberately designed in such a way that 20 simultaneous flows would saturate the network. The servers were configured to send application data as quickly as the DCCP protocol would allow. This is not indicative of normal network traffic but was done to ensure maximum network saturation and congestion occurred. Throughout the 12 hour period, the number of flows fluctuated between 5, 10, 15 and 20 simultaneous flows throughout the period. Each fluctuation lasted for a 20 minute duration before switching to the next fluctuation setting. During the 12 hour period a number of variables were measured. The service rate of the queue was constantly monitored to ensure the PBS mechanism calculation process did not hinder the maximum service rate of the outgoing interface. The added switching latency from the PBS mechanism was measured through simulation timers to determine how much impact the PBS mechanism had on the transfer throughput. CPU overheads and memory utilization were also measured to ensure the PBS mechanism is feasible in a moderately large network topology.

In addition to the specific variables explored above, the simulation was also monitored to make sure the BOD devices remained stable throughout the tested duration. The main method used to determine this was analysis of the throughput rates for the various flows that were recorded during the simulated period. Periods where throughput levels dropped were examined to ensure the reason for this drop was due
to empty queue levels, where there were no packets scheduled for transmission. In addition to this, all workstations were configured to halt the simulation if a session flow timeout event occurred. The final check for stability was provided by Opnet Modeler's animation recording facilities. These facilities allowed the flows between the simulated nodes to be watched in the form of animated packets which traveled between the nodes. The entire 12 hour duration was recorded and then played back at an increased speed to make sure there were no periods of inactivity and that packets arriving the BOD were transmitted out of the outgoing interface in a continual flow.

To ensure the results were valid, the experimentation was repeated on the three most complex network topologies that were created for use in this experimentation. These topologies are presented below in Figures 6.4 to 6.6.

![Network diagram](image)

Figure 6.4: First topology selected to test proposed mechanism.
Figure 6.5: Second topologies selected to test proposed mechanism.
6.10.2 Results

6.10.2.1 Queue depths and throughput

In all three tested topologies, the throughput rate experienced was either a product of the bandwidth available on the outgoing interface or the queue depth. There were no instances observed during the simulations where the calculations used in the mechanism resulted in a decrease in throughput. The computational requirement for the categorization of packets was found to take between 0.3 and 0.5 microseconds depending on the router configuration and hardware. For this reason, the throughput rate was never limited by this process. Where throughput rates dropped below the maximum throughput rate for a link, this was found to be caused by fluctuations in queue sizes on the outgoing interface. When the queue was empty there were drops
detected in the throughput rate, which was expected.

6.10.2.2 CPU overhead

Due to its minimalistic approach and the utilization of EIGRP metrics rather than calculating a number of the metrics independently, the PBS mechanism was found to have very little computational overhead. In order to determine the amount the CPU utilization increased as a result of the PBS mechanism, the 12 hour simulations were re-run and the categorization mechanism was disabled. The CPU utilization rates were then compared to determine what impact the PBS mechanism had on the CPU utilization. In all three topologies comparisons between the two results sets showed the results did not vary enough to suggest the variation was caused by the actions of PBS mechanism. While there was no relationship found between the mechanisms and the variance in results, it is worth noting that the maximum CPU utilization rates were found to be slightly higher where the PBS mechanism was enabled when compared to the results gathered where the mechanism was not enabled in all three topologies tested. In Topology 1, when the mechanism was activated, the maximum CPU utilization rate recorded was 82.401%, whereas, in the same topology when the mechanism was inactive, the maximum CPU utilization recorded was 78.113%. In Topology 2, when the mechanism was active, the maximum CPU utilization recorded was 79.892% which was higher than the 78.486% experienced in the same topology when the mechanism was deactivated. In the third topology, when the mechanism was active, the maximum CPU utilization value was reported at 80.553%, while, the maximum CPU utilization recording found when the mechanism was not active was 79.004%. This would indicate the PBS mechanism does have some impact on the maximum CPU utilization on the BOD. The CPU used for the simulation was the industry standard ISR G2 series processor found in Cisco 2900 series routers.

6.10.2.3 Stability and continuity

In all three topologies, the BOD devices remained stable and were able to operate in an environment where multiple BOD devices existed on the same network traffic
path. This information was gathered from throughput results which showed the flows configured between the various sender and receiver devices remained active for the entire 12 hour duration. To further verify this was the case, observation was carried out using Opnet Modelers animation viewer which showed graphically the transfer of data packets between the various devices for the entire 12 hour simulated period. Throughout the 12 hour periods, data flows between the various senders and receivers for the entire period were observed.

6.10.3 Analysis and discussion of results

In this experimentation, the PBS mechanism was placed into a medium sized network, where multiple BOD devices were configured and fluctuating traffic volumes were passed through these devices for a period of 12 hours (simulated). While the results derived from this experimentation are based largely on observation, from these observations, it is shown that the PBS mechanism proposed in this chapter is stable, scalable and has low overheads. The observations showed that the additional latency caused as a result of the PBS mechanism was between 0.3 and 0.5 microseconds. This value was a product of the hardware found on the BOD. In addition to this, CPU utilization levels were not found to be noticeably different to those that would be found when the mechanism was not present. Having stated this however, it should be noted iterations of the experiment where the PBS mechanism was active were found to have higher maximum CPU utilization values when compared to the same iteration where the mechanism was disabled.

Finally, the PBS mechanism was tested for stability by ensuring all flows were maintained as and when needed throughout the 12 hour period. Through the results collected and through observation of Opnet's Animation viewer, it was shown that the continuity of flows remained intact throughout the simulated period. These observations show that the PBS mechanism supports topologies with multiple inline BOD devices. The results also show the PBS mechanism is reliable and minimalistic in terms of CPU utilization and additional latency added to the data path.
6.11 Experiment 4: Effectiveness of the critical queue

The final experiment in this chapter explored the effectiveness of the critical queue by exploring how many of the packets sent via the critical queue were actually received within their useful lifespan. The reasons for doing this were to ensure the function of prioritizing these packets performed its intended function of aiding older packets in reaching their destination networks within the necessary time frame. If this was not achieved and the packets that were placed into the critical queue arrived at their destination stale on a consistent basis, it would bring into question the effectiveness and benefits of the PBS mechanism. In order to be effective the vast majority of packets placed into the critical queue must arrive at their destination within their valid lifespan window. The PBS mechanism’s ability to do this was measured in this experiment.

6.11.1 Experiment methodology

To carry out this experiment, the three topologies where the highest number of packets that were placed into the critical queue, out of the 20 topologies, were selected. The BOD was then modified in these three topologies to mark all the packets that were placed into the critical queue using the COS field in the IP header for that packet. All packets that were placed into the critical queue were given a COS value of 8. The workstations in the topology were then configured to detect the COS marking when incoming packets were received. When a packet was received with a COS value of 8 in the IP header, the workstation then calculated if the packet was fresh or stale. This was done by checking the current NTP time with the lifetime information in the DCCP header option field for that packet. If the packet was found to be fresh, a global fresh_pkt_rcvd_counter variable was incremented by one. Alternatively, if the packet was found to be stale, a global stale_pkt_rcvd_counter variable was incremented by one. In addition to these two variables, a third variable was also used which recorded the total number of packets placed into the critical queues on the BOD. This variable was called critical_pkt_sent_counter. This variable was used to ensure that all packets placed in the critical queue were received in either a stale state or a fresh state. Each time a packet was placed into the critical queue, the
critical_pkt_sent_counter value was incremented by one. To simplify the analysis of results, only one BOD was used on the topologies in which this experimentation was performed. Where the normal and discard queues were more frequently used, there was less contention for the preallocated bandwidth assigned to the critical queue. In such cases, the ratio of fresh packets being received that originated from the critical queue was higher because packets would have less queuing delay in the critical queue. In these topologies, packets placed in the critical queue were subjected to the longest delaying queue possible because of the higher number of packets categorized into this queue. The approach taken to create traffic profiles for the experiments was to design a worst case scenario type topology by selecting the three topologies where the critical queue was highly utilized. To achieve this, Topologies 14, 15 and 16 were used. In these topologies, a delay of 165 milliseconds was added to the links prior to the BOD device. This ensured a higher than normal number of packets were placed into the critical queue but not into the discard queue. The variation in the three topologies was created by altering the speed of the links after the BOD device. The simulations were run for a period of eight minutes. Collection of results was carried out after the first minute to ensure EIGRP convergence had occurred and CCID3 had stabilized. At the end of the eight minute period, the various global variables were recorded.

6.11.2 Results

Table 6.8 shows the global variables used to capture the results needed for the experiment. In particular, the results show the number of packets that were placed into the critical queue that were received fresh by the workstations. The results also show the number of packets that were placed into the critical queue that were received stale by the workstations. In addition, the results also show the total number of packets that were marked by the BOD as being placed in the critical queue for consistency purposes. Finally, the results show the percentage of packets that were placed into the critical queue that were received by the workstation within their useful lifespan.

In Table 6.8 it can be seen that in all the topologies the number of critical packets that were received fresh was significantly larger than the number of critical packets that were received stale. Specifically, between approximately 87 and 89 percent of
Table 6.8: Ratio of stale to fresh packets from critical queue that arrived on time.

<table>
<thead>
<tr>
<th>Topology</th>
<th>Total packets received fresh</th>
<th>Total packets received stale</th>
<th>Total packets reported by BOD</th>
<th>Percentage of packets arrived fresh</th>
</tr>
</thead>
<tbody>
<tr>
<td>14</td>
<td>6080</td>
<td>830</td>
<td>6910</td>
<td>87.988%</td>
</tr>
<tr>
<td>15</td>
<td>9470</td>
<td>1180</td>
<td>11150</td>
<td>89.417%</td>
</tr>
<tr>
<td>16</td>
<td>15350</td>
<td>1850</td>
<td>17150</td>
<td>89.232%</td>
</tr>
</tbody>
</table>

packets of transmitted from the critical queue were able to reach the final endpoint within their useful lifespan. This shows conclusively that the majority of the packets sent from the critical queue are received within their useful lifespan.

6.11.3 Analysis and discussion of results

While the results shown here indicate the PBS mechanism allows the majority of packets placed in the critical queue to be received within their useful lifespans, it must be noted that the network conditions were conducive to this occurring. It is also simply not possible to compare the results that occur when the categorization mechanism is enabled to results taken from a standard DCCP topology where categorization does not take place. This is because the act of prioritizing packets has a compounding effect on all other current and subsequent packets in that flow. If the actions of the PBS mechanism did not occur, then the congestion window for that flow may not increase or decrease in the same way as it would have when the PBS mechanism was active. This would affect the number of packets that were transported during the simulated period. For this reason, direct comparison between a standard DCCP topology and a topology where the PBS mechanism was enabled would not be valid because of the effects the PBS mechanism may have on flow dynamics.

6.12 Experimentation conclusion

The experimentation described in this chapter served two main purposes. The first purpose of the various experiments was to validate that the PBS mechanism worked effectively, reliably and in a stable manner, in a number of different topologies. The second main purpose of the experiments was to identify scenarios and network con-
ditions where the PBS mechanism would be most beneficial in improving DCCP performance. The results from the various experiments showed that both of these goals were achieved and that there is conclusive evidence supporting the potential benefit that can be obtained through the implementation of the PBS mechanism described in this chapter. If placed into networks where a high number of packets are becoming stale while in transit or where packets are nearing their maximum age when received at the destination, the PBS mechanism was shown to facilitate high levels of successful deliveries for packets placed into the critical queue. The experiments also showed the PBS mechanism was not as efficient in scenarios where the majority of packets it received were categorized into the normal queue and there was limited utilization of both the discard queue and the critical queue. In such scenarios, the performance experienced became comparable to typical DCCP performance. In terms of reliability, the networks tested remained stable and throughout the simulations all links and nodes remained operational. In these instances, the BOD devices remained stable and were able to service up to 20 simultaneous packet streams transferring data at rates governed only by the CCID3 protocol. Finally, the experiments showed that the probability based TNTCN value fell within the acceptable threshold of 10% variance and was able to produce improvement to overall network performance.

6.13 Limitations of the experimentation

One of the limitations of this experimentation was that the complexity of network needed to support the mechanism led to a situation where too many flows and variables existed in the topology to get an exact measure on what performance gains were generated through the mechanism.

Another issue that must also be stressed is that the performance gains were very much based on topology selected. During the analysis phase, the results showed categorically that the level of improvement, as well any adverse effects the PBS mechanism caused, became almost entirely dependent on the network topology. While efforts were made to generalize the findings to make them applicable to a large range of topologies, the results still should be treated as proof of concept rather than exactly what could be expected in any given topology.
An additional limitation with the experimentation, was that it did not take into consideration the impact EIGRP protocol had on the overall performance of the various flows. While EIGRP is designed to be as minimalistic as possible, there are overheads associated with its function that should be considered if the PBS is implemented. While there would be some impact caused through the addition of EIGRP needed by the mechanism, this impact is likely to be extremely minimal.

The final limitation with the experiments was that the workstations were designed in such a way that they generated packets as fast as the DCCP protocol would allow them to transmit. This is not what would be expected in real world network operations as the application layer would likely limit the rate of transfer, especially in the case of real time applications. In theory, the volume of traffic generated by the five to twenty servers in the simulation would be representative of a much larger number of real world devices than depicted in the experimental topology.

6.14 Chapter conclusion

This chapter has introduced a novel scheduling algorithm for the DCCP protocol that makes use of the additional option fields described in Chapter 4. Specifically, the chapter has introduced a PBS mechanism that utilizes an array of routing protocol information to predict the time that it is likely to take for packets to reach the destination network and proactively sorts and prioritizes packets based on this prediction. By placing packets into one of three queues, packets that are likely to become stale, or that have already become so, are pruned or given a lower priority to ensure they do not have an adverse effect on fresh packets utilizing the same contended resources.

Packets that need to be prioritized in order to avoid becoming stale are given the best chance of being able to be delivered on time. Finally, the PBS mechanism provides a deterministic queue that ensures the majority of normal packets remain largely unaffected by the actions occurring in the other two aforementioned queues.

Detailed discussion as to how the PBS mechanism should be implemented has been presented, and following this, a series of four experiments examining various elements of the PBS mechanism were described. These experiments showed that the PBS mechanism is stable, reliable and capable of offering benefit to CCID3 controlled
This chapter concludes, through the results obtained in the proof of concept implementation of the PBS mechanism, that not only is the mechanism workable, it is also beneficial in environments where there are high levels of stale packets. In situations where the number of stale packets is not significant, the design of the mechanism is such that the normal queue is utilized more frequently and performance levels become similar to those that would be experienced in standard DCCP queuing environments.
Chapter 7

Conclusion

7.1 Summary of the research

This chapter concludes the process of incorporating deadline scheduling into DCCP described in this thesis. The research has centered on the finite nature of real time data and showed how a packet’s lifespan information can be embedding into DCCP data packets. In addition to doing this, the research has also demonstrated a number of new scheduling mechanisms that make use of this information in order to improve DCCP performance when transferring real time data packets. This section provides a summary of the research described in this thesis.

Before deadline scheduling could be incorporated into DCCP, the DCCP standard was first modeled. In order to do this, the various RFC’s that make up the DCCP protocol were combined to form the standard DCCP model. Once this process was complete, the standard DCCP model was then implemented in Opnet Modeler. One of the unique characteristics of DCCP is that it does not retransmit lost packets, but does slow its sending rate down when packet loss occurs. Because of the infancy of the DCCP protocol, the effects this characteristic has on flow performance had not been extensively tested compared with TCP and UDP. The first contribution of the thesis was to investigate the effects placing fixed queue sizes on intermediate network devices had on DCCP performance. It was found that by employing small queues on intermediate devices between the DCCP sender and receiver, RTT times could be reduced for packets in a flow. This meant packets were received at a faster rate by the application. While RTT values were reduced, it was shown this reduction came at the expense of increased packet loss rates occurring in that flow as a result of queues
filling and not accepting new packets. When the fixed queue sizes were increased, there was a reduction in the number of dropped packets but also an increase in RTT values.

In addition to examining the effects fixed queue sizes had on DCCP performance, the experimentation also validated that the standard DCCP model used in the research operated in accordance with the various DCCP standards. It also provided a baseline of standard DCCP results and performance that later research made reference to. Once the standard DCCP model was found to be operating in accordance with the various RFC’s, the research then demonstrated how deadline scheduling could be incorporated into DCCP endpoints. To achieve this, two additional option fields were added to the DCCP data packet header. These option fields were configured to carry information relating to the lifespan of data being transported in that packet. In the first option field, the data’s maximum useful lifespan, as dictated by the sending application, was embedded. The second option was used to embed the NTP time when the packet was created. With these two option fields, intermediate BODs synchronized to the same NTP strata could then determine if packets were stale or fresh.

In addition to demonstrating how deadline scheduling could be implemented in DCCP, the performance degradation caused by doing this was also quantified through experimentation. It was found that the incorporation of the two additional option fields did add to the DCCP overheads and as a result impacted on DCCP performance. In particular, there was a decrease in goodput and throughput due to the added size of the DCCP-data packet header. The results did however show that the inclusion of the deadline scheduling did not cause any detriment to the rate at which DCCP reacted and/or recovered from packet loss events. As well as testing performance degradation caused by the inclusion of deadline scheduling, the experiments also tested DCCP stability after the deadline scheduling modification was added. It was found that the deadline scheduling options did not cause stability issues in environments where non deadline scheduling enabled devices were utilized. This meant the objectives of incorporating deadline scheduling into DCCP in a stable and backward compatible manner were met.

Once deadline scheduling was incorporated into DCCP, the next step taken was to configure intermediate BOD devices to utilize the packet lifespan information and
optimize DCCP performance using this information. In order to do this, two different approaches were explored in the research. In the first approach, intermediate devices used the lifespan information to detect stale packets. Once the device detected a stale packet, one of five mechanisms could be used to reduce the impact the stale packet had on the other packets in the queue. In the second approach, routing protocol information was used to predict whether a packet would reach its intended destination within its useful lifespan. Based on this, the packet was then placed into one of three queues to ensure it was given the best chance of delivery.

To expand on this, in the first approach five discard mechanism were introduced in theory. In addition to their advantages and disadvantages, the pseudo code used to implement the mechanisms was also shown. Following the theoretical overview of the mechanisms, four of the five mechanisms were then implemented into the simulated DCCP model and a series of proof of concept experiments were executed. The first objective of these experiments was to determine if the deadline scheduling mechanisms were able to perform better than a standard DCCP topology (where no discard actions took place) was able to perform. From the proof of concept experiments, it was found that all the discard mechanisms greatly increased goodput and RTT values for DCCP flows when placed into a severely congested network.

The second objective of the experimentation was to determine if the actions of the discard mechanism would cause fairness issues between competing flows. From the proof of concept modeling, it was concluded that all the discard mechanisms did result in fairness issues occurring when DCCP flows had different characteristics. In particular it was found flows with lower total delay values would become dominant over flows with higher total delay values as a result of the mechanism. The experiments concluded by exploring if a flow that was managed by a discard mechanism would perform less competitively than a similar competing flow that was not managed by a discard mechanism. From the results it was found that all the discard mechanisms, with the exception of the pure ECN model, did impact the flows competitiveness. The pure ECN model was found not to lose its competitiveness when competing with the non discard managed flow. It should be noted that the conditions used to measure fairness were extreme. The fairness issues would not be as significant in real world topologies.

In the second approach, a novel probabilistic scheduling technique was described
which made use of the deadline scheduling fields to optimize real time data transfers. Unlike the first approach which only reacted to stale packets after the fact, the PBS mechanism took a more proactive approach and estimated the likelihood a packet had of arriving at its destination within its useful lifespan. Based on this likelihood the packet was either prioritized to increase its likelihood of being delivered within its lifespan, placed into a normal queue, or placed into a discard queue so it did not cause detriment to non stale packets. To determine the packet’s likelihood of arriving on time, the EIGRP routing protocol was queried. Using various EIGRP metrics to determine the network conditions that occurred between the intermediate device and the final destination, the TNTCN variable was deduced. The research showed that the formula used to calculate the TNTCN was accurate for the purposes of the PBS mechanism. It was also shown that the PBS mechanism was not excessively computationally intensive and remained stable throughout long lived network transmissions. Of most importance however was the results which demonstrated the mechanism ensured the majority of packets placed in the critical queue (i.e. would likely have expired if no action was taken) were delivered within their useful lifespan time. The improvement to DCCP performance made possible by the PBS mechanism again validated the usefulness of incorporating deadline scheduling into DCCP and showed the benefits that can be achieved by intermediate devices that utilize the option fields it provides to optimize real time data transfers.

7.2 Contributions of this research

This section will summarize the main contributions that were achieved through this research. The research:

- Highlighted the need for the finite nature of real time data transfers to be considered when transporting real time data across modern networks.
- Created a standardized RFC6762 compliant DCCP model in DCCP.
- Quantified the effects of fixed queue sizes on DCCP performance.
- Described the variation in real-time application’s requirements.
Incorporated deadline scheduling into DCCP in a stable and backward compatible manner.

- Measured the effects incorporating deadline scheduling has on DCCP.
- Described and compared five packet discard mechanisms all of which are novel in DCCP.
- Described and tested a novel probabilistic packet scheduling mechanism for DCCP using CISCO's EIGRP protocol.

### 7.3 Implications of the research

This thesis has a number of implications for future research in both deadline scheduling and in DCCP in general. Firstly, this research adds to the body of evidence and supports the findings in [26, 29] that deadline scheduling improves the performance of real time transmissions. What is novel in this thesis is that this research tests deadline scheduling in DCCP and provides a baseline model of deadline scheduling enabled DCCP. The results have shown the newly defined deadline scheduling enabled DCCP model is stable and can be implemented directly into the DCCP with minimal disruption to DCCP performance. In addition, the use of options ensures the protocol is backward compatible with non deadline enabled DCCP devices. Having done this, future research can now explore additional scheduling techniques to those presented in this thesis.

Prior to this research there was limited incentive for application designers to choose DCCP over a more widely used protocol such as UDP. The biggest practical implication for DCCP achieved through this research is that the inclusion of deadline scheduling will make DCCP more attractive to application developers and hopefully this will result in DCCP being more widely utilized. The opportunities for new scheduling techniques that result from the inclusion of deadline scheduling may be the catalyst needed to make DCCP the future proof, advanced transport layer protocol of choice when developing real time applications.

In addition to making DCCP more attractive to application developers, most of the previous deadline scheduling research has focused on reducing the effects of pack-
ets once they had become stale. In addition to addressing this issue, the research described in this thesis has also introduced a novel scheduling and categorization mechanism that is based on predicting if a packet will become stale and taking necessary actions to reduce its impact based on this prediction. To do this, the routing protocol is used. Using the routing protocol in such a way highlights the need for cross layer cohesion and demonstrates how the transport layer can benefit from accessing information about the network from this source. Having proven the benefits of such a mechanism in DCCP in this research through proof of concept modeling, it is hoped that future research can adapt this mechanism to other transport layer protocol such as TCP and UDP.

7.4 Conclusion

In conclusion, the research described in this thesis represents significant progress towards the incorporation of deadline scheduling in DCCP. The ability to schedule packets on intermediate devices based on packet life information creates the possibility for a number of new scheduling techniques for DCCP and real time data that are not possible in the current DCCP standard. The research in this thesis shows a number of methods by which stale packets can be removed from networks to reduce congestion.

In addition, the thesis presents a novel PBS mechanism that calculates the probability packets have of reaching their destination and makes scheduling decisions based on this probability. This fundamental shift in improving how real time data is handled in modern networks is vital given the exponential increase in the rate of adoption of real time applications.
Appendix A

Topologies used in Chapter 6

This appendix will present the 20 topologies described in the Chapter 6 experiments.

The following settings are common to all experimentation topologies:

- Streams are measured from server to corresponding workstation.
  - Each stream is a simulated G.711 ulaw stream configured to send to packets as fast as the DCCP protocol permits.
  - Server_DCCP_1 will send traffic to Client_DCCP_1; Server_DCCP_2 will send traffic to Client_DCCP_2 and so on...
  - EIGRP is active on all routers.
  - No network load balancing occurs if multiple paths exist. No links fail during simulation.
  - A variety of link speeds are used to ensure variations in EIGRP metric values for realistic network topology conditions.
  - The Internet Protocol (IP) is used at layer 3 on routing devices.
  - Every router interface is configured on a unique IP subnet.
  - Subnets represented in the diagrams represent a unique IP sub network and are addressed accordingly.
Figure A.1: Chapter 6 experimental Topology 1.
Figure A.2: Chapter 6 experimental Topology 2.
Figure A.3: Chapter 6 experimental Topology 3.
Figure A.4: Chapter 6 experimental Topology 4.
Figure A.5: Chapter 6 experimental Topology 5
Figure A.6: Chapter 6 experimental Topology 6.
Figure A.7: Chapter 6 experimental Topology 7.

Figure A.8: Chapter 6 experimental Topology 8.
Figure A.9: Chapter 6 experimental Topology 9.

Figure A.10: Chapter 6 experimental Topology 10.
Figure A.11: Chapter 6 experimental Topology 11.

Figure A.12: Chapter 6 experimental Topology 12.
Figure A.13: Chapter 6 experimental Topology 13.

Figure A.14: Chapter 6 experimental Topologies 14, 15 and 16 (Link speeds varied for each iteration. (Slow; Med; Fast)).
Figure A.15: Chapter 6 experimental Topologies 17, 18 and 19 (Link speeds varied for each iteration (Slow; Med; Fast)).
Figure A.16: Chapter 6 experimental Topology 20.
Bibliography


