SESSION-LEVEL CONGESTION CONTROL FOR THE INTERNET

by

Siddharth Ramesh

A thesis submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Science

School of Computing
The University of Utah

May 2007
SUPERVISORY COMMITTEE APPROVAL

of a thesis submitted by

Siddharth Ramesh

This thesis has been read by each member of the following supervisory committee and by majority vote has been found to be satisfactory.

Chair: Sneha Kumar Kasera

John Carter

Jay Lepreau
To the Graduate Council of the University of Utah:

I have read the thesis of Siddharth Ramesh in its final form and have found that (1) its format, citations, and bibliographic style are consistent and acceptable; (2) its illustrative materials including figures, tables, and charts are in place; and (3) the final manuscript is satisfactory to the Supervisory Committee and is ready for submission to The Graduate School.

______________________________
Date

______________________________
Sneha Kumar Kasera
Chair: Supervisory Committee

Approved for the Major Department

______________________________
Martin Berzins
Chair/Director

Approved for the Graduate Council

______________________________
David S. Chapman
Dean of The Graduate School
ABSTRACT

Congestion caused by a large number of interacting Transmission Control Protocol (TCP) flows at a bottleneck network link is fundamentally different from that caused by a lesser number of flows sending large amounts of data. However, since existing congestion control schemes view congestion only from a packet-level perspective, they treat both to be the same, resulting in suboptimal per-flow performance.

The aim of this thesis is to develop session-level congestion control strategies for the Internet, to complement existing packet-level congestion control schemes. Two novel, search-based, session control mechanisms are presented that optimize for overall flow completion rate and per-flow performance respectively. The proposed session control mechanisms do not require any per-flow state or computation at the routers, make no assumption about input traffic characteristics and requirements, aim to avoid starvation of new flows when existing flows do not leave the system, and do not require any end host TCP modifications. Using evaluations under a wide variety of static and varying traffic load conditions, this thesis demonstrates the significant performance gains that session control provides.
To my entire family - mom, dad, brother, uncle, aunt and grandmom - for their constant motivation and prayers
CONTENTS

ABSTRACT ................................................................. iv
LIST OF FIGURES .......................................................... ix
LIST OF TABLES ........................................................... xiii
ACKNOWLEDGMENTS ....................................................... xiv

CHAPTERS
1. INTRODUCTION ....................................................... 1
   1.1 Contribution .................................................... 4
   1.2 Overview ...................................................... 4
2. MOTIVATING SESSION CONTROL ................................. 5
   2.1 Effect of the number of simultaneous TCP flows ............... 5
   2.2 Session control through rate-limiting .......................... 7
      2.2.1 Rate of flow completion .................................. 8
      2.2.2 Per-flow performance .................................... 9
      2.2.3 Fairness between small and large flows .................. 9
   2.3 Stability of flow completion rate .............................. 10
   2.4 Stability of flow completion times ............................ 13
   2.5 The effect of Pareto .......................................... 15
   2.6 Goals of our session control algorithm ....................... 15
3. SESSION CONTROL ALGORITHMS ................................. 18
   3.1 Algorithm 1 - Maximizing the flow completion rate ........... 18
   3.2 Algorithm 2 - Maximizing the per-flow performance ........... 22
4. PERFORMANCE EVALUATION AND DISCUSSION ................. 24
   4.1 Performance metrics ........................................... 24
   4.2 Experimental setup ........................................... 24
   4.3 No change in traffic pattern .................................. 25
      4.3.1 Increase in flow completion rate ......................... 25
      4.3.2 Increase in per-flow performance ....................... 27
      4.3.3 Increase in fairness between short and long flows ....... 29
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.3.4</td>
<td>Time behavior of flow admit and completion rates</td>
<td>29</td>
</tr>
<tr>
<td>4.4</td>
<td>Effect of change in traffic pattern</td>
<td>32</td>
</tr>
<tr>
<td>4.5</td>
<td>Changes in flow arrival rate</td>
<td>32</td>
</tr>
<tr>
<td>4.5.1</td>
<td>Flow completion rate</td>
<td>34</td>
</tr>
<tr>
<td>4.5.2</td>
<td>Per-flow performance</td>
<td>35</td>
</tr>
<tr>
<td>4.5.3</td>
<td>Fairness between short and long-lived flows</td>
<td>36</td>
</tr>
<tr>
<td>4.5.4</td>
<td>Time behavior of the session control algorithm</td>
<td>36</td>
</tr>
<tr>
<td>4.6</td>
<td>Periodic spikes in flow arrival rate</td>
<td>39</td>
</tr>
<tr>
<td>4.6.1</td>
<td>Flow completion rate</td>
<td>41</td>
</tr>
<tr>
<td>4.6.2</td>
<td>Per-flow performance</td>
<td>43</td>
</tr>
<tr>
<td>4.6.3</td>
<td>Fairness between short and long-lived flows</td>
<td>43</td>
</tr>
<tr>
<td>4.6.4</td>
<td>Time behavior of session control algorithm</td>
<td>43</td>
</tr>
<tr>
<td>4.7</td>
<td>Change in average flow size</td>
<td>45</td>
</tr>
<tr>
<td>4.7.1</td>
<td>Flow completion rate</td>
<td>48</td>
</tr>
<tr>
<td>4.7.2</td>
<td>Per-flow performance</td>
<td>48</td>
</tr>
<tr>
<td>4.7.3</td>
<td>Fairness between short and long-lived flows</td>
<td>48</td>
</tr>
<tr>
<td>4.7.4</td>
<td>Time behavior of our session control algorithm</td>
<td>48</td>
</tr>
<tr>
<td>4.8</td>
<td>Periodic spikes in the average flow size</td>
<td>52</td>
</tr>
<tr>
<td>4.8.1</td>
<td>Flow completion rate</td>
<td>53</td>
</tr>
<tr>
<td>4.8.1.1</td>
<td>High flow arrival rate</td>
<td>53</td>
</tr>
<tr>
<td>4.8.1.2</td>
<td>Low flow arrival rate</td>
<td>54</td>
</tr>
<tr>
<td>4.8.2</td>
<td>Per-flow performance</td>
<td>56</td>
</tr>
<tr>
<td>4.8.3</td>
<td>Fairness between short and long-lived flows</td>
<td>56</td>
</tr>
<tr>
<td>4.9</td>
<td>Change in bottleneck capacity</td>
<td>58</td>
</tr>
<tr>
<td>4.9.1</td>
<td>Increase in the available bandwidth</td>
<td>59</td>
</tr>
<tr>
<td>4.9.1.1</td>
<td>Flow completion rate</td>
<td>59</td>
</tr>
<tr>
<td>4.9.1.2</td>
<td>Per-flow performance</td>
<td>60</td>
</tr>
<tr>
<td>4.9.1.3</td>
<td>Fairness between short and long-lived flows</td>
<td>62</td>
</tr>
<tr>
<td>4.9.2</td>
<td>Decrease in the available bandwidth</td>
<td>62</td>
</tr>
<tr>
<td>4.9.2.1</td>
<td>Flow completion rate</td>
<td>62</td>
</tr>
<tr>
<td>4.9.2.2</td>
<td>Per-flow performance</td>
<td>63</td>
</tr>
<tr>
<td>4.9.2.3</td>
<td>Fairness between short and long-lived flows</td>
<td>65</td>
</tr>
<tr>
<td>4.10</td>
<td>Periodic spikes in bottleneck bandwidth</td>
<td>65</td>
</tr>
<tr>
<td>4.10.1</td>
<td>Flow completion rate</td>
<td>66</td>
</tr>
<tr>
<td>4.10.1.1</td>
<td>High flow arrival rate</td>
<td>66</td>
</tr>
<tr>
<td>4.10.1.2</td>
<td>Low flow arrival rate</td>
<td>67</td>
</tr>
<tr>
<td>4.10.2</td>
<td>Per-flow performance</td>
<td>69</td>
</tr>
<tr>
<td>4.10.3</td>
<td>Fairness between short and long-lived flows</td>
<td>69</td>
</tr>
<tr>
<td>4.10.4</td>
<td>Time behavior of the session control algorithm</td>
<td>69</td>
</tr>
<tr>
<td>4.11</td>
<td>Effect of long-lived flows</td>
<td>72</td>
</tr>
<tr>
<td>4.11.1</td>
<td>Flow completion rate</td>
<td>72</td>
</tr>
<tr>
<td>4.11.2</td>
<td>Per-flow performance</td>
<td>75</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

2.1 Simulation topology ................................................. 6
2.2 Effect of flow arrival rate on the service rate of the system ........ 7
2.3 Effect of flow admit rate on flow completion rate ................. 8
2.4 Stability of flow completion rate with and without session control ... 11
2.5 Time histogram of different goodputs. Top figure corresponds to zero
goodput per second. Bottom corresponds to a goodput of three or
more data packets per second ....................................... 12
2.6 Variation of the average flow completion time with time ............ 14
2.7 Effect of flow admit rate on flow completion rate with exponentially
distributed flow sizes ............................................. 16
4.1 Comparison of the number of flows completing when the input traffic
pattern is not changing with time. Flow arrival rate = 100 flows/s for
the top plot and 1000 flows/s for the bottom plot ................... 26
4.2 Histograms of flow durations with and without session control, when
the input traffic pattern is not changing with time .................... 28
4.3 Comparison of the average flow completion times when there the input
traffic pattern is not changing with time. Flow arrival rate is 1000
flows/s ........................................................... 29
4.4 Variation of flow admit and completion rates with time, when there
is no change in the input traffic pattern and no session control being
employed .......................................................... 30
4.5 Variation of flow admit and completion rates with time, when there
is no change in the input traffic pattern and $GSS + GA$ session control
algorithm is in use ................................................ 31
4.6 Variation of flow admit and completion rates with time, when there
is no change in the input traffic pattern and $CP$ session control
algorithm is in use ................................................ 32
4.7 Different load change scenarios considered for changes in the flow
arrival rate - $LOAD1$ (top) to $LOAD3$ (bottom). $FAR$ stands for
flow arrival rate .................................................... 33
4.8 Comparison of the number of flows completing when the input flow
arrival rate changes from 100 flows/s to 1000 flows/s ............... 34
4.9 Comparison of the average flow completion times when the input flow arrival rate changes from 100 to 1000 flows/s (top graph) and to 300 flows/s (bottom graph) ................................................................. 35

4.10 Comparison of the average size of flows completing when the input flow arrival rate changes from 100 to 1000 flows/s ................................. 36

4.11 Time variation of the flow admit and completion rates with no session control, for changes in the input flow arrival rate (Load3, FAR1=100, FAR2=1000) ................................................................. 37

4.12 Time variation of the flow admit and completion rates with CP session control algorithm, for changes in the input flow arrival rate (Load3, FAR1=100, FAR2=1000) ................................................................. 38

4.13 Time variation of the flow admit and completion rates with no session control (top) and with CP (bottom), for changes in the input flow arrival rate (Load2, FAR1=100, FAR2=300) ................................................................. 40

4.14 Periodic pulses in the flow arrival rate ........................................ 41

4.15 Comparison of the number of flows completing with periodic pulsations in the input flow arrival rate. Length of pulse is 10 seconds .... 42

4.16 Comparison of the number of flows completing with periodic pulsations in the input flow arrival rate. Length of pulse is 2 seconds .... 42

4.17 Comparison of average flow completion times with periodic pulses in the input flow arrival rate. Pulse length is 10 seconds for the top plot and 2 seconds for the bottom plot ................................................. 44

4.18 Variation of flow admit and completion rate with time when there are sharp pulses in the input flow arrival rate (Pulse length = 2 s, Interpulse duration = 2 s). The top plot corresponds to no session control, and the bottom plot corresponds to the CP session control algorithm ................................................................. 46

4.19 Different load change scenarios considered for changes in the average flow size - Load1 (top) to Load3 (bottom). AFS stands for average flow size ................................................................. 47

4.20 Comparison of the number of flows completing when the average flow size changes from 10000 to 30000 bytes. Flow arrival rate is 1000 flows/s for the top plot and 100 flows/s for the bottom plot .............. 49

4.21 Comparison of the average flow completion times of flows completed when the average flow size changes from 10000 to 30000 bytes. Flow arrival rate is 1000 flows/s for the top plot and 100 flows/s for the bottom plot ................................................................. 50

4.22 Time variation of the flow admit and completion rates with no session control, for changes in the average flow size (Load3, AFS1=10000, AFS2=30000) ................................................................. 51
4.23 Time variation of the flow admit and completion rates with CP session control algorithm, for changes in the average flow size (Load3, $AFS_1=10000$, $AFS_2=30000$) ........................................................................... 52

4.24 Periodic spikes in the average flow size ........................................ 53

4.25 Comparison of the number of flows completing, with periodic pulsations in the average flows size. Flow arrival rate is 1000 flows/s and length of pulse is 2 seconds. ................................................................. 54

4.26 Comparison of the number of flows completing, with periodic pulsations in the average flows size. Top plot corresponds to a pulse length of 10 seconds and the bottom plot corresponds to pulse length of 2 seconds. Flow arrival rate is 100 flows/s ........................ 55

4.27 Comparison of the average flow completion time, with pulsations in the average flows size. Top plot corresponds to a pulse length of 10 seconds and the bottom plot corresponds to pulse length of 2 seconds. Flow arrival rate is 100 flows/s ................................. 57

4.28 Different load change scenarios considered for changes in the available bottleneck bandwidth - Load1 (top) to Load2 (bottom) ................. 59

4.29 Comparison of the number of flows completing when the available bottleneck bandwidth increases from 10 Mbps to 20 Mbps. Flow Arrival Rate is 1000 flows/s .................................................. 60

4.30 Comparison of the number of flows completing when the available bottleneck bandwidth increases from 10 Mbps to 20 Mbps. Flow Arrival Rate is 100 flows/s ......................................................... 61

4.31 Comparison of the average flow completion times, when the available bottleneck bandwidth increases from 10 to 20 Mbps. Flow Arrival Rate is 1000 flows/s ................................................................. 61

4.32 Comparison of the average size of flows completed, with an increase in the available bottleneck bandwidth from 10 to 20 Mbps. Flow Arrival Rate is 1000 flows/s ................................................................. 62

4.33 Comparison of the number of flows completing, with a decrease in the available bottleneck bandwidth from 10 to 5 Mbps. Flow Arrival Rate is 1000 flows/s ................................................................. 63

4.34 Comparison of the number of flows completing, with a decrease in the available bottleneck bandwidth from 10 to 5 Mbps. Flow Arrival Rate is 100 flows/s ................................................................. 64

4.35 Periodic spikes in the available bottleneck bandwidth ..................... 66

4.36 Comparison of the number of flows completed with pulse-like variations in the available bandwidth. Flow arrival rate is 1000 flows/s and length of pulse is 2 seconds .................................................. 67

xi
4.37 Comparison of the number of flows completed with pulse-like variations in the available bandwidth. Flow arrival rate is 100 flows/s. Length of pulse is 10 seconds for the top plot and 2 seconds for the bottom plot ................................................................. 68

4.38 Comparison of the per-flow performance (indicated by the average flow completion time) with pulsations in the bottleneck bandwidth. Pulse length is 10 seconds for the top plot and 2 seconds for the bottom plot ................................................................. 70

4.39 Responsiveness of the CP session control algorithm to regular pulse-like variations in the bottleneck bandwidth ................................................................. 71

4.40 Comparison of the number of flows completing in the presence of extremely long-lived flows ................................................................. 73

4.41 Time variation of the flow admit and completion rates with CP session control algorithm, during the presence of extremely long-lived flows . . 74

4.42 Time variation of flow admit and completion rates with no session control, during the presence of extremely long-lived flows ............ 74

4.43 Comparison of the average flow completion time with and without session control, during the presence of extremely long-lived flows .... 75
LIST OF TABLES

2.1 Per-flow performance at different admit rates .......................... 9
2.2 Fairness between large and small flows with session control .......... 10
4.1 Average size of completed flows with no change in the traffic pattern . 30
4.2 Average size of completed flows with pulses in the flow arrival rate . . 45
4.3 Average size of completed flows with change in input flow size ........ 51
4.4 Average size of completed flows with pulsations in input flow size .... 58
4.5 Average flow completion times with decrease in bandwidth .......... 64
4.6 Average size of completed flows with decrease in bandwidth .......... 65
4.7 Average size of completed flows with pulses in bottleneck bandwidth . 71
ACKNOWLEDGMENTS

This thesis is by far the biggest academic work in my life, and it would not have been possible without the support of many many people.

First and most of all, I thank my advisor Dr. Sneha Kumar Kasera profusely for his immense support through the last two years. His motivation and confidence in me early on, when I started on my master’s assignment, was what led me to read more about networking problems and start liking networking research. I have tremendously enjoyed and learnt from the numerous discussions we have had, not only with regard to this thesis, but also on other topics in computer networks and security. Though his support in this thesis cannot be understated in any way, his role was much more than just that - right from shepherding me in the initial period of my grad life and helping me get accustomed to the academic way of life here, to training me to sharpen my writing and presentation skills. Thanks a lot!

I would like to thank my committee members Prof. John Carter and Prof. Jay Lepreau for leading me in the right direction, making me dig deeper and understand many aspects of the problem better. I also thank them for their support whenever needed.

I thank Prof. Rajeev Balasubramonian and Prof. John Carter for allowing access to the Corvus cluster of machines, which helped exponentially decrease the run time of my experiments.

I thank my fellow graduate students Sachin Goyal, Prashanth Radhakrishnan and Jun Cheol Park for patiently listening to and engaging in many insightful discussions on various aspects of my thesis. I have learnt a lot from each of them on a variety of systems topics. I also thank Jun for his help with the ns2 network simulator and in helping me review the code I had written.
Thanks are also due to all the professors who have taught me and all TAs who have helped me.

I thank all my friends for the last two years for making graduate life an extremely enjoyable and memorable experience.

And of course, no words are enough to thank my family, who have supported me all the way through. I dedicate this thesis to them.
CHAPTER 1

INTRODUCTION

In the current Internet, congestion is controlled only at the packet level. There is no mechanism to prevent the build up of a large number of flows at a router. The existing congestion control schemes use packet drops at the routers in conjunction with source back-off to control the number of packets sent towards a congested router link. They neither prevent a large number of flows from contending with each other, nor block new flows from joining the network and further increasing congestion. As a result of this imbalance in the granularity of congestion control, when a large number of flows gain admission into the network due to the absence of any flow-level control at routers, each flow obtains very little bandwidth and consequently, stays in the network for a long time. This hurts the flow completion times of elastic applications and the interactivity of multimedia applications. Furthermore, it has been shown in [24, 19, 25] that as the number of simultaneous flows increases to a high value, TCP’s Additive Increase Multiplicative Decrease (AIMD) algorithm is no longer fair in sharing the bandwidth at a congested link. This is due to the increase in the packet drop rate which further reduces the throughput of flows from their fair share. There is a significant degradation in network goodput as well, because of the increased retransmissions.

This thesis addresses the problem of controlling congestion at the session (or flow) granularity. The solution we propose is to complement, rather than substitute, packet-level congestion control mechanisms currently in place. As in the case of packet level congestion control, an effective session level control must involve dropping sessions at the routers in conjunction with session back-off at the sources. However, in this thesis, we focus only on session level control at the routers.
The overall functional goals of our session-level control at the router are to maximize per-flow performance of elastic TCP flows, improve network stability, and improve fairness between short and long-lived flows, during session overload at routers. In addition to these functional goals, our session level control also meets the following logistical and engineering requirements to be practical and deployable.

- Routers do not maintain any per-flow state.
- No a priori traffic (or flow) distribution is assumed and no characterization of incoming flows (such as using token bucket parameters, peak rate etc.) is required.
- Our scheme is robust to changes in traffic patterns, both sudden and gradual.
- In the worst case, the performance does not degrade below that obtained without any session control.

Traditionally, overload control is achieved by comparing one or more measures of system or network load against one or more specified load thresholds. For example, there could be a certain minimum throughput guarantee for each flow; all flow admission decisions would then be based on whether this throughput assurance can be met for the incoming as well as existing flows. The limitation of such threshold-based approaches is that, the specified thresholds, unless tuned, are not likely to result in optimal performance under all network and system conditions. In fact, the performance could be well below optimal when network and system conditions change significantly.

This thesis proposes novel search-based session control strategies that do away with load thresholds. The bottleneck router controls the rate at which new sessions are admitted into the network during session overload. The router searches through a range space of admit rates and selects the rate that optimizes one of

\footnote{We use the term session interchangeably with flow so as not to confuse flow-level control with flow control.}
two differing goals - maximizing the overall flow completion rate and maximizing per-flow performance.

The first algorithm aims to select the optimal admit rate that maximizes the flow completion rate at all times. We use a unique combination of Golden Section Search (GSS) and Gradient Ascent (GA) to perform this search. GSS is a powerful optimization algorithm that does not require the calculation of derivatives of the function it is optimizing. It is quick to reduce the search space considerably. However, we find that using only GSS causes fluctuations about the optimal value as the range of search gets smaller. For this reason, our scheme shifts to GA as the optimal value is neared. GA is a simple hill climbing optimization algorithm that approaches the local maxima by taking steps proportional to the gradient of the function at the current point, until a peak is found. GA moves more steadily towards the optimal value and is less affected by noise and oddities in the measurement of flow completion rates.

We find that maximizing the flow completion rate does not translate to maximizing the per-flow performance. This is because, for Pareto distributed input flow sizes, maximal flow completion rates are achieved at admit rates higher than it. On the other hand, per-flow performance is maximized only when the number of flows in the system does not increase with time. Hence we develop a second algorithm whose primary goal is to maximize per-flow performance. Constrained by this goal, the secondary goal of the algorithm is also maximize flow-completion rate. The router searches for the maximum admit rate that does not exceed the completion rate. The algorithm starts by admitting all incoming flows. It then reduces the admit rate to bring it as close as possible to the flow completion rate. This reduction is multiplicative and proportional to the difference between the admit and the completion rates. The algorithm also continuously probes for extra bandwidth and changes in network conditions by increasing the flow admit rate and checking for a corresponding increase in the flow completion rate.

The flow completion rate and hence the optimal admit rate are not fixed. These could change with variations in the flow arrival rate, flow size and available
bandwidth. Both the algorithms detect such changes and converge to the new optimal admit rate. This makes our search-based schemes extremely adaptable to changing traffic patterns and network conditions.

We evaluate our session control scheme using extensive ns2 [27] simulations under a variety of stationary and changing traffic loads and find that our scheme results in significant improvements in per-flow performance and network fairness.

1.1 Contribution

The major contributions of this thesis can be summarized as

1. A demonstration of the value of controlling congestion at the session level in addition to packet level.

2. Novel session control mechanisms for the Internet.

3. A thorough evaluation of the above schemes under a variety of static and varying network loads.

1.2 Overview

The rest of this thesis is organized as follows. Chapter 2 motivates the need for session control. We also discuss some of the important aspects of session control, that lead to the two session control algorithms detailed in Chapter 3. We follow it up with a detailed evaluation of our algorithms under a variety of static and varying load scenarios in Chapter 4. Chapter 5 discusses some of the deployment scenarios and issues involved with implementing such a scheme at the router. Chapter 6 reviews other recent work on admission control, specially for elastic flows, and discusses their drawbacks and limitations. We conclude in Chapter 7 with directions for future work.
CHAPTER 2
MOTIVATING SESSION CONTROL

In this chapter we motivate our approach by identifying the drawback with current network systems which do not employ any session control. We validate this inference with the help of simulations leading to our two search-based algorithms.

A network system at the session (or flow) granularity can be thought of as a queuing system - new flows arrive into the input queue (connection establishment), get serviced (data transfer) and then leave the system (connection termination). Since all flows that gain admission into the system, start their transfers immediately without waiting, the number of servers in the network queuing system is the equal to the number of simultaneous flows sharing the system. As is expected, the serving rate of each server is inversely proportional to the number of servers in the system - each flow receives proportionally lesser throughput when many flows share the bottleneck resources.

2.1 Effect of the number of simultaneous TCP flows

When the number of simultaneous flows increases to a large value, the per-flow throughput is no longer proportionally less - it is worse. This, as we briefly mentioned in Chapter 1, is because TCP’s AIMD algorithm no longer emulates Processor Sharing (PS) when the number of competing flows grows to a large value [19, 25]. When the number of TCP flows sharing a link is large (greater than the number of packets in the delay-bandwidth product [19]), the ability of TCP to share the bottleneck link efficiently and fairly decreases. Qiu et al. [25] state that when this happens, only as many connections as the number of packets the network can hold, are active, or achieve goodput considerably larger than zero;
the rest of the connections are practically shut-off due to constant timeouts. This implies that as the flow arrival rate increases, the average service rate (and hence, the average goodput) of the flows decreases to even less than the fair share.

We perform a series of simulations using the ns2 simulator to test this hypothesis. The simulation network topology consists of a single bottleneck topology (of bandwidth 10 Mbps and delay 100 ms) as shown in Figure 2.1. There are 10 sources connected to the bottleneck through access links, each of which has a bandwidth of 10 Mbps and delays uniformly randomly varying between 10 and 50 ms. The 10 sources generate new file transfer TCP connections according to a Poisson process. The flow (file) sizes are drawn from a Pareto distribution with a shape parameter of 1.5 and mean of 10000 bytes (typical of web traffic [6]).

The mean of the Poisson distribution used to generate connections at the sources is varied, thereby controlling the rate at which new flows arrive at the bottleneck router. This in turn, affects the number of simultaneous flows sharing the bottleneck. Figure 2.2 plots the effect of the rate of flow arrival on the service rate of the system. The flow arrival rate has a huge impact on the service times - for a given network setting (bottleneck bandwidth) and traffic pattern (average flow size), there is an optimal arrival rate at which the number of flows completing in a given interval of time (i.e., the network service rate) is maximized.

For an arrival rate greater than this optimal value, the number of simultaneous TCP connections becomes too high and the service rate is affected in a negative

![Figure 2.1. Simulation topology](image_url)
way as explained above - most of the connections are constantly shut off and only a very less number of connections are able to achieve good throughput and complete. For an arrival rate less than the optimal value, the network is under-utilized leading to low service rates.

2.2 Session control through rate-limiting

To motivate the need for session control at routers, we perform another experiment in which the mean arrival rate of new flows is fixed at 1000 flows per second, which corresponds to a highly suboptimal arrival rate in Figure 2.2. However, the router is engineered to admit flows only at a fixed rate in each experiment. Flows are admitted (or discarded) by allowing (or dropping) TCP SYN packets at the bottleneck link. Flows that are dropped retry using the usual TCP retransmit mechanism. We now discuss the effect admission control has on different system parameters.
2.2.1 Rate of flow completion

Figure 2.3 shows the rate of completion of flows as a function of the rate at which flows are admitted. As can be seen, the flow admit rate greatly affects the flow completion rate.

For a given network setting and average flow size, the flow completion rate (≡ number of flows completing in a given interval of time) is maximized for a particular admit rate (≈ 250 flows/s). For any other admit rate, the number of flows completing is suboptimal. Further, for very high and low flow admit rates, the flow completion rate is very low compared to the optimal number. We note that when no session control is deployed and all the flows are admitted as they arrive (this corresponds to the value 1000 on the x-axis), the flow completion rate is less than half the maximum.

![Figure 2.3. Effect of flow admit rate on flow completion rate](image-url)
2.2.2 Per-flow performance

Table 2.1 shows the per-flow performance (average flow completion times and goodputs) of the flows that complete, for different admit rates at the router.

As is observed from the table, as we decrease the rate of flow admission, both the average flow completion times and goodputs improve. This is obvious, since a lower rate of admission corresponds to lesser number of simultaneous flows sharing the bottleneck, and hence, a greater per-flow share.

2.2.3 Fairness between small and large flows

The flow sizes of the generated flows, as we said earlier, are drawn from a Pareto distribution with a mean of 10000 Bytes and shape parameter of 1.5. For these parameters, the mean corresponds to 81 percentile of the flow sizes; i.e., 81% of the flows are less than 10000 Bytes in size. Table 2.2 lists the average size of the flows that complete, for different rates of flow admission. The table also lists the percentile corresponding to 10000 bytes among the completing flow sizes.

With no session control (corresponding to an admit rate of 1000 flows/s), the distribution of the flows that complete is highly skewed, compared to the input traffic distribution. Almost all (≈ 99%) flows that complete are small flows (less than 10000 B). However, with session control through rate-limiting, the output

<table>
<thead>
<tr>
<th>Flow admit rate (flows/s)</th>
<th>Flow completion rate (flows/s)</th>
<th>Avg flow completion time (s)</th>
<th>Average goodput (Kbps)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>43.944</td>
<td>69.178</td>
<td>2.61</td>
</tr>
<tr>
<td>500</td>
<td>54.88</td>
<td>67.606</td>
<td>2.64</td>
</tr>
<tr>
<td>350</td>
<td>76.508</td>
<td>58.118</td>
<td>3.25</td>
</tr>
<tr>
<td>250</td>
<td>107.088</td>
<td>42.031</td>
<td>4.46</td>
</tr>
<tr>
<td>200</td>
<td>106.036</td>
<td>35.327</td>
<td>5.79</td>
</tr>
<tr>
<td>150</td>
<td>102.6</td>
<td>25.001</td>
<td>8.73</td>
</tr>
<tr>
<td>100</td>
<td>99.84</td>
<td>4.053</td>
<td>49.83</td>
</tr>
<tr>
<td>65</td>
<td>64.716</td>
<td>3.0178</td>
<td>63.48</td>
</tr>
<tr>
<td>40</td>
<td>39.852</td>
<td>3.32</td>
<td>64.24</td>
</tr>
</tbody>
</table>
distribution follows the input flow distribution much more closely - for an admit rate of 100 flows/s, 10000 B corresponds to almost 81 percentile, as should be.

Thus, even if we are not able to control the rate at which new flows arrive at a bottleneck router, just by deploying session control and admitting flows at an optimal rate, we can significantly improve the flow completion rate, per-flow performance and the fairness between 'mice' and 'elephant' flows.

2.3 Stability of flow completion rate

If the rate of flow arrival is greater than the rate of flow completion, we would expect the rate of flow completion to progressively decrease to a very low value. This is because, as the number of flows in the system increases, the per-flow performance keeps decreasing to negligibly low values. However, we found that this is not the case - in the absence of any changes to the network setting, the flow completion rate remains fairly constant with time even when the number of flows in the system keeps increasing. In Figure 2.4, we plot the time behavior of flow completion rate without and with (at 250 flows/s) rate limiting. As can be seen, even without any session control, the flow completion rate remains quite stable (≈ 50 flows/s) with time. With session control (at 250 flows/s), the stability region is much higher (≈ 110 flows/s).

<table>
<thead>
<tr>
<th>Flow admit rate (flows/s)</th>
<th>Avg. size of completed flows (Bytes)</th>
<th>Percentile corresponding to 10000 B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1000</td>
<td>4657.86</td>
<td>98.63</td>
</tr>
<tr>
<td>500</td>
<td>4775.58</td>
<td>97.98</td>
</tr>
<tr>
<td>350</td>
<td>5225.34</td>
<td>95.55</td>
</tr>
<tr>
<td>250</td>
<td>6534.13</td>
<td>88.21</td>
</tr>
<tr>
<td>200</td>
<td>7491.14</td>
<td>84.27</td>
</tr>
<tr>
<td>150</td>
<td>8478.90</td>
<td>82.05</td>
</tr>
<tr>
<td>100</td>
<td>10142.48</td>
<td>80.98</td>
</tr>
<tr>
<td>65</td>
<td>10171.74</td>
<td>81.14</td>
</tr>
<tr>
<td>40</td>
<td>9719.72</td>
<td>81.23</td>
</tr>
</tbody>
</table>

Table 2.2. Fairness between large and small flows with session control
The stability of the flow completion rate with time can be explained by observing the time varying histogram of different goodputs in the system as in Figure 2.5. The top plot shows the variation of the number of flows receiving zero goodput per second, with time. The bottom plot shows the same for flows being able to send three or more data (1000 byte) packets every second.

As can be seen from these plots, the number of flows that receive zero goodput increases linearly with time, whereas the number of flows that are able to send one or more packets stays approximately constant with time. This suggests that more and more flows get successively ‘backed-off’ and are practically ‘shut-off’, while a constant number of flows are able to achieve reasonable goodput. This results in the stability of the flow completion rate. This observation is also in tune with results from literature [25].
Figure 2.5. Time histogram of different goodputs. Top figure corresponds to zero goodput per second. Bottom corresponds to a goodput of three or more data packets per second.
2.4 Stability of flow completion times

Flow Completion Time (FCT) is defined as the total time that a flow takes to complete sending its data (from sending its first SYN packet to receiving its last FIN). In a system with admission control, this includes a waiting component and a data transfer component. The waiting component is the time taken by the flow to successfully transmit its SYN packet and establish the connection. The data transfer time is the time needed to send all data once the connection has been established.

We saw in the previous section that even if the number of competing flows in the system is increasing, the flow completion rate reaches a stable value and remains approximately constant with time. This, as we argued, is because most of the competing flows get successively 'backed-off' and only a small set of flows (let us call this set of flows the 'useful set') are able to achieve a reasonable throughput at any point of time.

However, the same stability argument does not hold true for the flow completion times (or flow durations) of the flows that complete. The number of flows in the system has an adverse impact on the flow durations. This is because, while flow completion rate is not affected by what kind of flows complete, the average flow completion time depends greatly on whether the flows that complete are 'old' flows (that have stayed in the system for a long time) or 'new' flows (that have just gained admission). If most of the flows that complete are 'old' flows, the average flow completion time would be much higher than if most of the completing flows are 'new' flows.

We plot (in Figure 2.6) the time-variation of the average FCT, for different degrees of session control. When there is no session control, the average FCT increases with time. The same kind of plot is observed for an admit rate of 250 flows/s since the admit rate is still greater than the flow completion rate (Figure 2.3). However, when the admit rate is 100 flows/s (corresponding to a completion rate of 100 flows/s), the average FCT remains fairly constant throughout (apart from some spikes in the later part of the graph). This is because the number
of flows in the system does not increase with time and the ‘useful set’ is replaced by predominantly ‘new’ flows only (though the spikes correspond to more ‘old’ flows entering the system with time).

As can be seen from Table 2.1, though the highest flow completion rate is achieved for an admit rate of 250 flows/s, the average flow completion time is still high ($\approx 25$ s). There is a sharp reduction (increase) in the average flow completion time (average goodput) for an admit rate of 100 flows/s (which also corresponds to an average flow completion rate of 100 flows/s). In this experiment, 100 flows/s is the maximum admit rate that equals the completion rate. In other words, this is the highest admit rate at which the number of competing flows in the system does not increase. Since there is always a constant number of flows in the system, the system is in a state of stable equilibrium and the per-flow performance (throughput and flow completion times) is maximized. Admit rates less than 100 flows/s also correspond to good per-flow performance, since the admit rate equals the completion rate.
However, since the completion rate is bound by the admit rate, the system is not fully utilized resulting in lesser number of flows completing.

### 2.5 The effect of Pareto

Another interesting observation needs to be made from Figure 2.3. From this figure, we note that the maximum flow completion rate ($\approx 110$ flows/s) is achieved at a much higher admit rate ($\approx 250$ flows/s). This appears counter-intuitive since, even if the system reaches steady-state (as discussed in the previous section), the number of flows in the system is building continuously, putting the system under more and more strain. However, this phenomenon can be attributed to the Pareto nature of flow sizes. As mentioned in the experimental setup, sizes of flows are Pareto distributed (with a mean of 10000 and shape parameter of 1.5). Examining the mean of our Pareto distribution, we find that it corresponds to 81 percentile of flow sizes. This implies that a large number of flows have a flow size below the mean and complete much faster (within a few round trip times) maintaining a high overall flow completion rate. To corroborate this hypothesis, we repeat the above experiment with all the same parameters except for the flow size, which are drawn from an Exponential distribution (instead of Pareto) with the same average. Figure 2.7 shows the effect of flow admit rate on the flow completion rate for this experiment. It can be seen that the peak of the graph corresponding to maximum flow completion rate ($\approx 100$ flows/s) corresponds to an admit rate of approximately 100 flows/s. Contrasting this with Figure 2.3, we confirm our hypothesis that the reason for the admit rate being higher than the maximum flow completion rate, is the Pareto nature of the incoming traffic.

### 2.6 Goals of our session control algorithm

Summarizing the observations from the previous sections,

1. Irrespective of the admit rate, the flow completion rate stabilizes to a nearly constant value, in the absence of other changes to the network setting and traffic characteristics.
2. For Pareto flow size distributions (which is characteristic of web traffic), the flow completion rate is maximized at an admit rate that is higher than it.

3. The per-flow performance (throughput and flow completion times) is maximized for any admit rate that is less than or equal to the flow completion rate since there is no buildup of flows in the system.

The above observations lead to two different goals for our session control algorithm:

- maximizing the flow completion rate, and
- maximizing the per-flow performance

In this thesis, we develop two algorithms that, respectively, optimize either of these goals.

Having motivated the need for session control, in the upcoming chapters we detail our session control algorithms, address some important issues such as how
a router determines the optimal admit rate and how it adapts the optimal with changes in traffic patterns.
CHAPTER 3

SESSION CONTROL ALGORITHMS

The two goals of the session control algorithm were outlined in the previous chapter - maximizing flow completion rate and maximizing per-flow performance. As we saw from the initial experiments, both these goals are tangential - when the flow completion rate is maximized, the per-flow performance is not optimal, and vice versa. Hence, we develop two different algorithms which separately aim to optimize either of these goals. We evaluate both these algorithms thoroughly in the next chapter and compare the performance of the two.

3.1 Algorithm 1 - Maximizing the flow completion rate

The goal of this algorithm is that it must find the admit rate that maximizes the flow completion rate. Further, since the maximum flow completion rate can shift with time (corresponding to changes in traffic patterns and background traffic), the algorithm needs to detect such changes and converge to the new maximum in a reasonable amount of time. The problem of finding the maximum (minimum) is typical of any global-optimization algorithm that maximizes (minimizes) a function - the “function” in this case being the rate of flow completion. In our context, several practical issues make the optimization problem more challenging. First, the “function” in our problem changes with time even if evaluated at the same point, i.e., $f_t(x)$ need not the same as $f_{t-1}(x)$. Further, the number of flows completing during a time interval is not just a function of the number of flows admitted during that interval, but depends on all previous flow admit rates, i.e., if the admit rate during some time interval $i$ is $a_{t_i}$, the flow completion rate $d(t_i)$ is not just a function of $a_{t_i}$, but a function of $(a_{t_i}, a_{t_{i-1}}, a_{t_{i-2}}, \ldots)$. This is because the number of flows
completing during any interval is a function of the number of flows sharing the system. This, in turn, depends on all the flow admit rates in the past, and not just the one during the current time interval.

In order to search for the optimal admit rate that maximizes the flow completion rate under changing conditions, we develop a new heuristic. Our heuristic uses a combination of two search algorithms - Golden Section Search and Gradient Ascent. Golden Section Search (GSS) is a powerful optimization algorithm that is especially attractive for our context because it does not require the calculations of derivatives of the function it is optimizing. It is commonly used in many practical real world applications such as evaluation of electric power tariff at power plants [14], optimizations in spreadsheets [12], and in controller designs [28]. GSS finds the maximum (or minimum) of a function by successively narrowing the bounds containing the maximum (minimum) until a certain tolerance level is reached. This algorithm requires an initial bracketing interval \((a, b, c)\) such that \(a < b < c\) and \(f(a) < f(b) > f(c)\). Starting with this bracketing, the function is evaluated at another point \(x\) which is located at a golden fraction \((0.3819)\) into the larger interval. The golden ratio defines the most efficient way to bracket an interval. In other words, the optimal bracketing interval \((a, b, c)\) has its middle point \(b\) a fractional distance \(0.3819\) from one end and \(0.618\) from the other end. Successive points are chosen inside the existing bracketing using this ratio. The bracketing is then readjusted based on the function value at the chosen point. This narrowing of the bracketing continues until the range of the bracket shrinks to lesser than a threshold. GSS is quick to reduce the search space considerably, but we find that using only GSS causes a lot of fluctuations about the optimal value as the range of bracketing gets smaller. This is because as the search approaches the optimal value, the slope of the graph becomes flatter, implying that neighboring admit rates have very close flow completion rates. Since GSS jumps on either side of the optimal value, it becomes very sensitive to the noise in function measurement. The effect of this noise does not tend to be significant when the optimal value is far away, since the difference in flow completion rates are then quite distinct. To remove
these fluctuations, our scheme shifts to the Gradient Ascent algorithm when the bracketing range is smaller than a threshold. Gradient Ascent (GA) is a simple hill climbing optimization algorithm that approaches the local maxima by taking steps proportional to the gradient of the function at the current point, until a peak is found. Since the gradient becomes smaller as the peak is approached, the step size also becomes smaller. Thus GA moves more steadily towards the optimal value and is much less affected by noise and oddities in the measurement of the flow completion rate. Our scheme is detailed in Algorithm 1.

<table>
<thead>
<tr>
<th>Algorithm 1</th>
<th>Algorithm to maximize flow completion rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\text{SHIFT}(a, b, c, d) : (a) \leftarrow (b); (b) \leftarrow (c); (c) \leftarrow (d)$</td>
<td>$\text{GOLD} \leftarrow 1.618$</td>
</tr>
<tr>
<td>$\alpha_{\text{min}} \leftarrow \text{Minimum flow admit rate threshold}$</td>
<td>$C \leftarrow 0.3819$</td>
</tr>
<tr>
<td>$\alpha_t \leftarrow \text{No of flows allowed during time interval } t$</td>
<td>$d_x \leftarrow \text{No. of flows departed when using admit rate of } x$</td>
</tr>
<tr>
<td><strong>Ensure:</strong> $\alpha_t \geq \alpha_{\text{min}}$</td>
<td></td>
</tr>
<tr>
<td>//Golden Space Search to reduce the search range</td>
<td></td>
</tr>
<tr>
<td>$\alpha_1 \leftarrow \alpha_{\text{min}}$</td>
<td></td>
</tr>
<tr>
<td>$\alpha_2 \leftarrow \alpha_1 + \text{stepSize}$</td>
<td></td>
</tr>
<tr>
<td>while $d_{\alpha_{t-2}} \leq d_{\alpha_{t-1}} \leq d_{\alpha_t}$ do</td>
<td></td>
</tr>
<tr>
<td>$\alpha_{t+1} \leftarrow \alpha_t + \text{GOLD} \ast (\alpha_t - \alpha_{t-1})$</td>
<td></td>
</tr>
<tr>
<td>end while</td>
<td></td>
</tr>
<tr>
<td>$a \leftarrow \alpha_{t-2}; b \leftarrow \alpha_{t-1}; c \leftarrow \alpha_t$</td>
<td></td>
</tr>
<tr>
<td>while $c - a \geq \text{GSS_THRESHOLD}$ do</td>
<td></td>
</tr>
<tr>
<td>$x \leftarrow b + C \ast (c - b)$</td>
<td></td>
</tr>
<tr>
<td>if $d_x &gt; d_b$ then</td>
<td></td>
</tr>
<tr>
<td>$\text{SHIFT}(a, b, x, b + C \ast (c - b))$</td>
<td></td>
</tr>
<tr>
<td>else</td>
<td></td>
</tr>
<tr>
<td>$\text{SHIFT}(c, x, b, x - C \ast (x - a))$</td>
<td></td>
</tr>
<tr>
<td>end if</td>
<td></td>
</tr>
<tr>
<td>end while</td>
<td></td>
</tr>
<tr>
<td>//Gradient Ascent</td>
<td></td>
</tr>
<tr>
<td>while TRUE do</td>
<td></td>
</tr>
<tr>
<td>$\alpha_{t+1} \leftarrow \alpha_t + \frac{d_{\alpha_t} - d_{\alpha_{t-1}}}{\alpha_t - \alpha_{t-1}} \ast \text{stepSize}$</td>
<td></td>
</tr>
<tr>
<td>end while</td>
<td></td>
</tr>
</tbody>
</table>
In brief, our algorithm performs the following actions.

1. **Brackets the maximum:**
   When the algorithm starts, it must first bracket the peak before GSS takes over. Bracketing of the peak is done in a straightforward manner - the admit rate is successively increased until there is a drop in the flow completion rate, at which point a bracket has been found.

2. **Narrows the bracketing range:**
   Using Golden Section Search, the algorithm successively divides the bracketing interval according to the golden ratio, i.e., the next point chosen is a fraction 0.3819 into the larger interval. Even if the initial bracketing does not conform to the golden ratio, dividing the bracket successively using this ratio quickly converges the bracketing towards conformance. GSS continues to reduce the bracketing range until it becomes lesser than a specified threshold ($GSS_{\text{THRESHOLD}}$ in Algorithm 1) at which point, it starts to use GA.

3. **Moves steadily towards the peak:**
   GA tends towards the peak more slowly than GSS, taking steps whose length is proportional to the gradient of the function at the current point. The actual step size also depends on a fixed step size ($stepSize$ in Algorithm 1) with which the gradient is multiplied.

4. **Detects changes in traffic pattern:**
   The algorithm remains in GA unless it detects a sudden change in the traffic pattern (using a heuristic which we explain ahead). GA shifts automatically with small and smooth changes in the traffic pattern, thus keeping the algorithm very responsive. However, when the traffic change is sudden and significant, we find that GA does not converge quickly enough to the new peak. Hence, we enhance our heuristic such that, when the algorithm detects such sudden changes, it loops back to GSS and incorporates the faster procedure to localize the region around the new peak. Our algorithm maintains a sliding
window of average flow completion rates that serves as its short-term memory. When there is a continual decrease in the average flow completion rate over the length of the window, and the decrease is greater than a threshold, the algorithm infers a sudden traffic change, and shifts back to GSS$^1$.

The $\alpha_{min}$ threshold defines the minimum accessibility of the network link to new flows. It is introduced to prevent the starvation of new flows in gaining admission, when existing flows are long lived and do not leave the network. We defer more discussion on this issue until Chapter 4. The convergence time of the above algorithm is linear in the range space since successive points are chosen linearly with additional function evaluations.

### 3.2 Algorithm 2 - Maximizing the per-flow performance

The primary goal of this algorithm is to maximize per-flow performance. As with the previous algorithm, there might be changes in the network settings and input traffic characteristics which might affect the flow completion rate. The algorithm has to be robust enough to detect such changes and automatically adapt to them. As we observed in the previous chapter, per-flow performance (flow completion time and throughput) is maximized when the flow admit rate is less than or equal to the flow completion rate. This is because, there is no buildup of flows inside the system, and hence, the per-flow share of the bottleneck is maximized. Thus when the admit rate is greater than the completion rate, the algorithm has to decrease it to bring it close to the completion rate. We refer to this stage of the algorithm as the 'closen' phase of the algorithm.

While closening remains the primary goal of this algorithm, a secondary goal of this algorithm is also to maximize the number of flows which complete. Note that the admit rate always acts as an upper bound for the completion rate. As a

---

$^1$Note that none of the the parameters used are load threshold parameters, as those used in threshold-based admission control schemes. These are used solely for the convergence of the algorithm and are independent of the traffic pattern. Hence our algorithm does not lose its robustness by employing these thresholds.
result, there could be many (very low) admit rates at which the completion rate equals it. However, at such rates the system still performs suboptimally since the bottleneck bandwidth is not being used to its full capacity. Hence the goal is to find the maximum flow admit rate that equals the flow completion rate. When there is an increase in the available bandwidth of the bottleneck (due to a decrease in the background traffic), the system can admit and sustain more number of simultaneous flows. In order to detect such increases in the available bandwidth, the second stage of this algorithm is to ‘probe’ for additional bandwidth.

We use a Multiplicative Increase Multiplicative Decrease (MIMD) algorithm in order to ‘close’ and ‘probe’. Our algorithm is detailed in Algorithm 2. The decrease factor is proportional to the difference between the flow admit and completion rates. We prefer a multiplicative increase (with a constant increase factor) over additive increase (AI), since AI would lead to a slower convergence to the optimal value (slower probing), when the admit rates are sufficiently high. Furthermore, the algorithm smoothen the measured flow completion rate using a weighted moving average to prevent oscillations due to bursty traffic.

Algorithm 2 Algorithm to maximize per-flow performance

\( \alpha_{\text{min}} \leftarrow \) Minimum flow admit rate threshold
\( \alpha_t \leftarrow \) No. of flows allowed during time interval \( t \)
\( a_t \leftarrow \) No. of new flows arriving during time interval \( t \)
\( d_x \leftarrow \) No. of flows departed when using admit rate of \( x \)

Ensure: \( \alpha_t \geq \alpha_{\text{min}} \)

//Smoothen flow completion rate estimate with history
\( SFCR \leftarrow (1 - \beta) \ast SFCR + \beta \ast d_{\alpha_t} \)
\( \alpha_{t+1} \leftarrow \alpha_t \)

if \( \alpha_t > SFCR \ast BUFFER \_ FACTOR \) then
    //Multiplicative Decrease
    \( \alpha_{t+1} \leftarrow \alpha_t + \text{stepSize} \ast \frac{SFCR - \alpha_t}{SFCR} \)
else
    // Multiplicative Increase
    \( \alpha_{t+1} \leftarrow SFCR \ast INCREASE \_ FACTOR \)
end if
CHAPTER 4

PERFORMANCE EVALUATION AND DISCUSSION

In this chapter, we evaluate both our session control algorithms with the help of simulations using the ns2 simulator.

4.1 Performance metrics

We mainly concentrate on two metrics of performance - flow completion rate (or equivalently, the number of flows completing in an interval) and per-flow performance (indicated by average flow completion times and goodputs). As we had seen in the Chapter 2, when the flow size distribution is Pareto, maximizing the flow completion rate is not the same as maximizing per-flow performance. In some experiments where it is necessary to show the functioning of our algorithm under changing traffic patterns, we also show the time behavior of the algorithm and its convergence to a new optimal admit rate.

4.2 Experimental setup

The simulation topology is as explained in Chapter 2, Figure 2.1. The bottleneck router buffer is set (in terms of packets) as a product of the bandwidth and the average delay of all flows passing through it. The stepSize and GSS.THRESHOLD used in Algorithm 1, are set to 20 and 100 respectively. The epoch between successive admit rate evaluation for Algorithm 1 is 2 seconds. For Algorithm 2, stepSize is set to 40, BUFFER.FACTOR to 1.05 and INCREASE.FACTOR to 1.1. The epoch between successive admit rate evaluation for this algorithm is 1 second.
In order to evaluate both our session control algorithms, we have extended the Queue/DropTail class in ns2 to a DropTailSessionControlQueue class which functions as a drop-tail queue enhanced to implement both our session control algorithms. We perform several experiments under a variety of network conditions. We describe these experiments and their results in the following sections. In all forthcoming plots, we refer to Algorithm 1 as 'GSS+GA' and Algorithm 2 as 'CP' (for Close and Probe).

4.3 No change in traffic pattern

In this set of experiments, the traffic is kept stationary such that the optimal admit rate does not vary with time. We observe two important properties of our session control algorithm - convergence and optimality.

4.3.1 Increase in flow completion rate

Figure 4.1 shows a comparison of the number of flows completing during the simulation run for four cases: no session control, optimal session control, session control using our GSS+GA algorithm and session control using our CP algorithm, for two different average flow sizes (10000 and 30000 bytes). The optimal admit rates and the corresponding flow completion rates are obtained by trying a large range of admit rates using a brute force method. The top plot corresponds to a flow arrival rate of 100 flows/s and a simulation time of 500 seconds. This flow arrival rate corresponds to no session overload for an average flow size of 10000 bytes and corresponds to session overload for the higher flow size. The bottom plot corresponds to a flow arrival rate of 1000 flows/s - this constitutes session congestion for both the file size distributions. The simulation time in this case is 350 seconds.

There are four important observations to be made from Figure 4.1. First, when there is session congestion (bottom plot), there is a huge improvement in the number of flows completing when there is session control as opposed to no session control. Especially, when the average flow size is 30000 bytes, and the flow arrival rate is
Figure 4.1. Comparison of the number of flows completing when the input traffic pattern is not changing with time. Flow arrival rate = 100 flows/s for the top plot and 1000 flows/s for the bottom plot.
1000 flows/s, the number of flows completed is \( \approx 17 \) times more when using either of our session control scheme. Second, \( CP \) performs almost as well as optimal session control under all traffic loads. Even though \( GSS + GA \) performs much better than no session control, it does not perform better than \( CP \) in any scenario. Third, when there is no session-level congestion (top plot), neither session control algorithm performs any worse than having no session control. The last observation is that, with no session control, the number of flows completing in a given time (and hence, the flow completion rate) is heavily dependent on the flow arrival rate - the flow completion rate reduces from 100.22 (50111 flows in 500 s) to 46.04 (16115 flows in 350 s) when the input arrival rate changes from 100 to 1000 flows/s. However, there is no such effect when session control is in place - flow completion rate using \( CP \) is 100.19 (50095 flows in 500 s) with a flow arrival rate of 100 flows/s, and 102.98 (36043 flows in 350 s) with a flow arrival rate of 1000 flows/s. In fact, this is proof that the session control algorithm is doing a good job removing the effect flow arrival rate has on performance.

### 4.3.2 Increase in per-flow performance

As mentioned earlier, the goal of the \( CP \) Session Control algorithm is to maximize the per-flow performance by making sure that the number of simultaneous flows in the system remains fairly constant with time. \( GSS + GA \) can also be expected to improve per-flow performance over the no session control case, since lesser number of flows are competing for the bottleneck share. Figure 4.2 shows the smoothened histogram of flow durations with and without session control. The average flow size used is 10000 bytes and the average flow arrival rate is 1000 flows/s.

We observe that, with session control, a very high fraction of the completing flows have very small flow completion times as opposed to no session control. Note that this improvement is in addition to the increase in the total number of completing flows due to session control. Further, comparing our two algorithms, it is clear that the improvement is much more for \( CP \) than for \( GSS + GA \) - a much
Figure 4.2. Histograms of flow durations with and without session control, when the input traffic pattern is not changing with time.

A higher fraction of flows complete with a small flow completion time with $CP$. This is in tune with the goal of the $CP$ algorithm to improve per-flow performance.

Figure 4.3 shows the average flow completion times for a flow arrival rate of 1000 flows/s. Note that the ‘optimal’ average flow completion in this figure is not the best average flow completion time obtainable; it merely refers to the flow completion time corresponding to the maximal flow completion rate. The average flow completion time is dependent only on the number of simultaneous flows and can be made arbitrarily small by admitting lesser and lesser flows into the system. However, this would have an adverse effect on the flow completion rate. We can observe from Figure 4.3 that during session congestion, session control (especially, $CP$) improves the per-flow performance by huge amounts.
4.3.3 Increase in fairness between short and long flows

In addition to the above benefits, as discussed earlier, session control also increases the fairness between short and long flows in the system.

Table 4.1 compares the average flow size of all flows completed during the same interval, with and without session control. Though the average flow size is approximately the same when the system is under no load (≈ 10000 Bytes, 100 flows/s), when the level of session congestion increases, the disparity in fairness becomes more obvious - the average size of flows completed is up to two times more with CP. Once again, we see that CP does a better job as compared to GSS+GA.

4.3.4 Time behavior of flow admit and completion rates

In order to study the working of our algorithms, we now plot the variation of the flow admit and flow completion rates for no session control, and session control using GSS+GA and CP. Instead of plotting the same experiment for all the three
Table 4.1. Average size of completed flows with no change in the traffic pattern

<table>
<thead>
<tr>
<th>Avg Flow Size Generated (Bytes)</th>
<th>Flow Arrival Rate (flows/s)</th>
<th>Avg Flow Size Completed (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No SC</td>
<td>GSS+GA</td>
</tr>
<tr>
<td>10000</td>
<td>100</td>
<td>9921.55</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>4646.12</td>
</tr>
<tr>
<td>30000</td>
<td>100</td>
<td>16248.64</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>11972.16</td>
</tr>
</tbody>
</table>

different algorithms, we choose the experiments individually for each case such that the plots are descriptive to analyze the working of the algorithm. For no session control and $CP$, we plot experiments using a flow arrival rate of 1000 flows/s and an average flow size of 10000 bytes. For $GSS + GA$, we choose an experiment with an average flow arrival rate of 100 flows/s and an average flow size of 30000 bytes.

Figure 4.4 plots the time-line behavior of when no session control is being employed at the router. Note that the even though the flow admit rate is much higher than the completion rate, the completion rate stabilizes at around 50 flows/s.

![Figure 4.4. Variation of flow admit and completion rates with time, when there is no change in the input traffic pattern and no session control being employed](image-url)
Figure 4.5 plots the variation of the admit and completion rates with time, when session control using $GSS + GA$, is in place. The initial part of the $GSS + GA$ algorithm (shown enlarged in the embedded figure) shows the Golden Section Search part of the algorithm. After sufficient reduction in the search space, Gradient Ascent takes over and the variation in the admit rate is smoother (though in this figure, the variation during the GA phase is higher than normal). Note that even $GSS + GA$ admits at a rate higher than the completion rate, in an attempt to maximize the flow completion rate.

Figure 4.6 plots the behavior of the $CP$ algorithm with time. The flow admit rate plot is almost indistinguishable from the flow completion rate plot, hinting that the algorithm is doing a good job in finding the maximum admit rate closest to the completion rate. Also note the regular spikes in the admit rate that correspond to the algorithm trying to probe for additional bandwidth to admit more flows to check if the completion rate increases proportionally.

![Figure 4.5](image-url)

**Figure 4.5.** Variation of flow admit and completion rates with time, when there is no change in the input traffic pattern and $GSS + GA$ session control algorithm is in use.
4.4 Effect of change in traffic pattern

We also experiment with a variety of changing traffic patterns and network conditions to observe how our session control algorithm reacts to such changes. We vary three network and traffic parameters - flow arrival rate, average flow size and bottleneck capacity. For each of these three broad categories, we vary the rate at which the parameters are changing from sudden to gradual.

4.5 Changes in flow arrival rate

Even though we have argued (and shown) that the input flow arrival rate does not have any effect on the optimality of the flow admit rate, it is interesting to see how our algorithms react to sudden changes in input load caused by an increase in the arrival rate of new flows. We perform various experiments with varying degrees of load change, including periodic spikes in the input rate.

Figure 4.7 shows the different rates of load changes that we consider - from gradual (top plot, LOAD1) to sudden (bottom plot, LOAD3). The experiments are each run for 300 seconds.
Figure 4.7. Different load change scenarios considered for changes in the flow arrival rate - LOAD1 (top) to LOAD3 (bottom). FAR stands for flow arrival rate.
Two different values of \((FAR_1, FAR_2)\) (from Figure 4.7) are considered - (100, 1000) and (100, 300). The first set represents going from underload (100 flows/s) to high overload (1000 flows/s) and back to underload. The second constitutes a jump from underload to low overload and back. The average flow size was maintained at 10000 bytes (distributed Pareto).

### 4.5.1 Flow completion rate

Figure 4.8 compares the number of flows completed using different algorithms (including no session control) for the load scenarios depicted in Figure 4.7. We see that, more flows complete with session control than without it. Further, between both our session control algorithms, \(CP\) performs better than \(GSS + GA\) even with respect to the number of flows getting completed.

![Figure 4.8](image.png)

**Figure 4.8.** Comparison of the number of flows completing when the input flow arrival rate changes from 100 flows/s to 1000 flows/s
4.5.2 Per-flow performance

Figure 4.9 compares the per-flow performance of flows completed (indicated by the average flow completion times), under the various load setups mentioned. Under all the load-change scenarios considered, the average flow completion time obtained using the $CP$ algorithm is the lowest. Another observation to be made is that the benefit of $GSS + GA$ is more pronounced at higher flow arrival rates.

![Comparison of average flow completion times](image)

**Figure 4.9.** Comparison of the average flow completion times when the input flow arrival rate changes from 100 to 1000 flows/s (top graph) and to 300 flows/s (bottom graph)
4.5.3 Fairness between short and long-lived flows

As we showed in section 4.3.3, controlling the number of sessions also increases the fairness between short and long lived flows in the system. Here we compare the average sizes of the flows that complete under changing input load conditions. Figure 4.10 compares the average sizes of all flows that complete. As expected, the average flow size is maximum using the CP session control algorithm.

4.5.4 Time behavior of the session control algorithm

We plot the time varying behavior of the CP session control algorithm and compare it with the no session control case to emphasize the importance of employing session control.

Figure 4.11 plots the flow admission and completion rates as a function of time when no session control is used. The load-change condition is that of Load-3 (Figure 4.7) - there is a sudden increase in the flow arrival rate from 100 flows/s to 1000 flows/s at time $t = 125$ s. The flow arrival rate jumps back to 100 flows/s at $t = 175$ s.

![Graph showing average flow size comparison](image)

**Figure 4.10.** Comparison of the average size of flows completing when the input flow arrival rate changes from 100 to 1000 flows/s
Figure 4.11. Time variation of the flow admit and completion rates with no session control, for changes in the input flow arrival rate (Load3, FAR1=100, FAR2=1000)

As can be observed from the figure, for the first 125 seconds, the flow completion rate is fairly stable and close to the flow arrival rate ($\approx 100$ flows/s), indicating no session overload. However, from 125 to 175 seconds, the admit rate shoots to very high values $^1$ due to the increase in flow arrival rate and the lack of any session control. During this time interval, corresponding to this increase in flow admit rate, there is a dip in the flow completion rate. When the flow arrival rate returns to session underload after $t = 175$s, the flow completion rate also returns to a value of around 100 flows/s. The dependence of the rate of flow completion on the rate of flow arrival when there is no session control, is obvious from this plot.

Figure 4.12 plots the flow admit and completion rates for the same load scenario considered above, but when employing our $CP$ session control algorithm. Note that through out the run time of the experiment, the flow admit rate closely follows the

$^1$The admit rate is still lower than the arrival rate (of 1000 flows/s) even when there is no session control, because of the size limitation of the router packet buffer, which causes some SYN packets to be dropped because of packet congestion
flow completion rate. Even when there is a spurt in the flow admit rate at $t \approx 125$ s, it is only because of an increase in the flow completion rate. The increase in the flow admit rate does not affect the flow completion rate in any negative way - the flow completion rate continues to remain greater than 100 flows/s. As a result of the flow admit rate not exceeding what the bottleneck can 'sustain' at any point of time, the flows which do get admitted get good throughput, which is reflected in the low average flow durations seen earlier.

Another important observation to make from Figure 4.12 is that during the initial period (0 - 40 s), the flow completion rate is less than the steady value, and reaches the stable value (of $\approx 100$ flows/s) only at around 40 seconds. The reason for this phenomenon is that, as noted in Algorithm 2 (CP), the flow completion rate is first smoothened using its history. During the first few seconds, there are not enough flows in the system and the flow completion rate is low. Since the history

![Figure 4.12](image)

**Figure 4.12.** Time variation of the flow admit and completion rates with CP session control algorithm, for changes in the input flow arrival rate (Load3, $FAR_1=100$, $FAR_2=1000$)
has also not built up, the smoothened value of the flow completion rate takes some
time to rise up to the steady value. This causes a degradation of the performance
of this algorithm during this period. However in practice, this would occur only
when the router is rebooted or when there are periods of prolonged drought in the
flow arrival rate (temporary droughts would be taken care of by the flows already
present in the system), and hence do not affect the performance of the algorithm
under normal circumstances.

Similarly, we compare the time behavior when the flow arrival rate changes
from 100 to 300 flows/s. The top graph in Figure 4.13 plots the flow admission
and completion rates as a function of time when no session control is used. The
load-change condition is that of Load-2 (Figure 4.7) - there is a gradual increase in
the flow arrival rate from 100 flows/s to 300 flows/s across 45 s (from t = 80 s to t
= 125 s). The flow arrival rate decreases at the same rate starting at t = 175 s.

The flow admit rate starts increasing at t = 80 s and increases to greater than
250 flows/s. When this happens, there is a dip in the flow completion rate to about
100 flows/s. Flow completion rate once again reaches the previous value (of ≈ 125
flows/s) after the overload ceases. In contrast, when \( CP \) is employed (bottom plot
in Figure 4.13), there is no decrease of the flow completion rate because of the
increase in flow arrival rate. The completion rate once again closely follows the
admit rate. Not only is the number of flows completing higher, on an average flows
complete about twice as fast as well (Figure 4.9).

4.6 Periodic spikes in flow arrival rate

In this section, we experiment with periodic pulses in the flow arrival rate and
the effect it has on the flow completion rate and per-flow performance.

Figure 4.14 shows the variation we introduce in the flow arrival rate. The
experiments are run for 200 seconds each, and in each experiment, the pulsations
start at t = 50 s, varying the flow arrival rate from 100 to 1000 flows/s. We also
vary both, the length of the pulse as well as the time interval between consecutive
Figure 4.13. Time variation of the flow admit and completion rates with no session control (top) and with CP (bottom), for changes in the input flow arrival rate (Load2, FAR1=100, FAR2=300)
pulses and observe how they affect performance. The average flow size is 10000 bytes (Pareto distributed) and the bottleneck bandwidth is 10 Mbps.

4.6.1 Flow completion rate

Figure 4.15 compares the number of flows completing during the experiment run for different interpulse durations. Each pulse is 10 seconds long. As can be seen, the number of flows completed is always greater with session control than without it. More importantly, note that the difference in performance with and without session control decreases as the time interval between pulses increases (from \( \approx 8000 \) when the interpulse duration is 2 seconds, to \( \approx 1500 \) when the interpulse duration is 50 seconds). The reason for this phenomenon is that, when the interpulse duration is small, the total time spent by the system under session overload is higher than when the interpulse duration is large. As a result, the effect of session control is much more appreciable for smaller interpulse durations.

Figure 4.16 compares the number of flows completing for a pulse length of 2 seconds. We observe the same phenomenon we observed earlier (for 10 second

\[ \begin{array}{c|c|c|c}
\text{Interpulse duration} & \text{Pulse length} \\
\hline
\text{FAR1} & 50 & \text{FAR2} \\
\hline
\text{Time (s)} & \text{Flow Arrival Rate (flows/s)} & \text{Pulse length} & \text{Interpulse duration}
\end{array} \]

**Figure 4.14.** Periodic pulses in the flow arrival rate
**Figure 4.15.** Comparison of the number of flows completing with periodic pulsations in the input flow arrival rate. Length of pulse is 10 seconds.

**Figure 4.16.** Comparison of the number of flows completing with periodic pulsations in the input flow arrival rate. Length of pulse is 2 seconds.
pulses) - the performance improvement due to session control is more when the total time spent in overload is higher.

It is to be noted that the reason for the smaller improvement with lesser session overload is not because our session control algorithm performs any worse with smaller overload. In fact, when the interpulse duration is the same, the number of flows completed with \( CP \) remains approximately the same for pulse lengths of 2 and 10. However, the no session control case performs much worse when the pulse length is 10 seconds (because of longer periods of overload). This demonstrates the resilience of our session control algorithm to sudden and sharp changes in load conditions.

### 4.6.2 Per-flow performance

Figure 4.17 compares the per-flow performance of completed flows, for the different load scenarios. As is expected, \( CP \) performs the best with the smallest flow completion times among the three. An interesting observation is that, in most cases, having no session control has better flow completion times than \( GSS + GA \) session control algorithm. This is because, \( GSS + GA \) does not do a very good job adapting to fast changing loads. As a result, even during underload, not all incoming flows are admitted, unnecessarily delaying some flows.

### 4.6.3 Fairness between short and long-lived flows

Table 4.2 shows the average sizes of the flows that complete, with and without session control. As expected, session control improves the overall average of all flows completing. Further, \( CP \) does a better job compared to \( GSS + GA \) (which only performs marginally better than no session control when the pulse length is 2 seconds).

### 4.6.4 Time behavior of session control algorithm

We now plot the time variation of the flow admit and completion rates. The top plot of Figure 4.18 shows the variation when no session control is in place. Both the
Figure 4.17. Comparison of average flow completion times with periodic pulses in the input flow arrival rate. Pulse length is 10 seconds for the top plot and 2 seconds for the bottom plot.
Table 4.2. Average size of completed flows with pulses in the flow arrival rate

<table>
<thead>
<tr>
<th>Pulse Length (s)</th>
<th>Interpulse Duration (s)</th>
<th>Avg Flow Size Completed (Bytes) No SC</th>
<th>GSS+GA</th>
<th>CP</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>2</td>
<td>6607.47</td>
<td>6935.65</td>
<td>9135.85</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>6644.12</td>
<td>6726.72</td>
<td>8865.20</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>7388.32</td>
<td>7815.23</td>
<td>8909.62</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>6676.21</td>
<td>6784.38</td>
<td>9370.86</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>6612.99</td>
<td>8569.50</td>
<td>9166.75</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>6774.37</td>
<td>8528.52</td>
<td>9040.24</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>7280.65</td>
<td>8592.69</td>
<td>8711.19</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>6727.59</td>
<td>8260.53</td>
<td>9139.34</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>6669.09</td>
<td>6723.30</td>
<td>9286.02</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>6600.19</td>
<td>8124.75</td>
<td>9075.67</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>6828.10</td>
<td>8255.09</td>
<td>9012.42</td>
</tr>
</tbody>
</table>

Pulse length and interpulse duration are 2 seconds. This corresponds to roughly half the total time (from 50 seconds onwards) being spent in rapidly changing session overload. As can be seen, the flow admit rate oscillates sharply corresponding to sharp changes in the incoming flow rate. As a result, after the oscillations start, the system performs suboptimally. The average flow completion rate drops from 100.14 flows/s during the first 50 s (before the pulses start) to 81.35 flows/s during the next 150 s (after the pulses start). The bottom plot shows the corresponding plot for CP algorithm. We can observe that the oscillations in the flow arrival rate has no effect on both the admit and completion rates.

4.7 Change in average flow size

In this section, we vary the average flow size of the input traffic, the distribution still being Pareto.

Figure 4.19 shows the different load scenarios considered. The experiment is run for 300 seconds. The average flow size is changed from 10000 bytes to 30000 bytes (i.e., AFS1 is 10000, AFS2 is 30000). We experiment with two flow arrival rates - 100 flows/s and 1000 flows/s.
Figure 4.18. Variation of flow admit and completion rate with time when there are sharp pulses in the input flow arrival rate (Pulse length = 2 s, Interpulse duration = 2 s). The top plot corresponds to no session control, and the bottom plot corresponds to the CP session control algorithm.
Figure 4.19. Different load change scenarios considered for changes in the average flow size - Load1 (top) to Load3 (bottom). AFS stands for average flow size.
4.7.1 Flow completion rate

Figure 4.20 shows the number of flows completing when there is an increase in the average flow size, for all the load-change scenarios mentioned earlier. The top plot corresponds to a flow arrival rate of 1000 flows/s, and the bottom plot, 100 flows/s. There is a huge increase in the number of flows completed (upto a factor of 2.2) with session control. Also, it is worth noting that for high flow arrival rates, $GSS + GA$ out-performs $CP$, as it rightly should.

4.7.2 Per-flow performance

Figure 4.21 compares the per-flow performance as indicated by the average flow completion time. Using the $CP$ session control algorithm produces huge benefits in the average flow completion times. $GSS + GA$ performs worse than no session control in some cases, once again proving that maximizing the flow completion rate is not the same as maximizing per-flow performance.

4.7.3 Fairness between short and long-lived flows

Table 4.3 compares the average size of all flows completed. As expected, $CP$ performs the best with average sizes more than two times that without any session control, when the flow arrival rate is 1000 flows/s. There is an improvement (though not as significant) when the average flow arrival rate is 100 flows/s also.

4.7.4 Time behavior of our session control algorithm

We observe the time behavior of our algorithms when there is a change in the average flow size, in order to better understand the convergence and optimality properties of our algorithms. Since 1000 flows/s already corresponds to high session overload, its time plot would not give any new insights into convergence when the system moves from underload to overload. Hence, we only show the time plots for flow arrival rate of 100 flows/s.

Figure 4.22 plots the flow admit and completion rates as a function of time, when no session control algorithm is employed. The plot corresponds to $Load3$
Figure 4.20. Comparison of the number of flows completing when the average flow size changes from 10000 to 30000 bytes. Flow arrival rate is 1000 flows/s for the top plot and 100 flows/s for the bottom plot.
Figure 4.21. Comparison of the average flow completion times of flows completed when the average flow size changes from 10000 to 30000 bytes. Flow arrival rate is 1000 flows/s for the top plot and 100 flows/s for the bottom plot.
Table 4.3. Average size of completed flows with change in input flow size

<table>
<thead>
<tr>
<th>Flow Arrival Rate (flows/s)</th>
<th>Load type</th>
<th>Avg Flow Size Completed (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>No SC</td>
</tr>
<tr>
<td>1000</td>
<td>Load1</td>
<td>5695.45</td>
</tr>
<tr>
<td></td>
<td>Load2</td>
<td>5039.56</td>
</tr>
<tr>
<td></td>
<td>Load3</td>
<td>4758.30</td>
</tr>
<tr>
<td>100</td>
<td>Load1</td>
<td>12113.00</td>
</tr>
<tr>
<td></td>
<td>Load2</td>
<td>12000.09</td>
</tr>
<tr>
<td></td>
<td>Load3</td>
<td>11557.53</td>
</tr>
</tbody>
</table>

Figure 4.22. Time variation of the flow admit and completion rates with no session control, for changes in the average flow size (Load3, AFS1=10000, AFS2=30000) (Figure 4.19) with the average flow size abruptly changing from 10000 to 30000 at t = 150 s. Until t = 150 s, the completion rate is close to the admit rate. However at t = 150 s, the system enters session overload and there is a sharp decrease in the flow completion rate due to the three-fold increase in the average flow size. However, the admit rate continues to remain the same since there is no session control.
Figure 4.23 plots the time variation of the flow admit and completion rates when the $CP$ session control algorithm is in use. Notice the flow admit rate closely follows the flow completion rate throughout the run time. At $t = 150$ s, when there is a sharp dip in the flow completion rate, the admit rate also drops to the new optimum value ($\approx 50$ flows/s).

During session overload (i.e., the last 150 seconds), the average flow completion rate with no session control is 39.20 flows/s, and the average flow completion rate with $CP$ is 46.12 flows/s - there is not a significant difference in the total number of flows completing (Figure 4.20). However, as was seen earlier, there is a significant difference in the per-flow performance (Figure 4.21).

### 4.8 Periodic spikes in the average flow size

We introduce pulsations in the average flow size of input traffic (Figure 4.24) and study the effect on the flow completion rate as well as per-flow performance.

![Figure 4.23. Time variation of the flow admit and completion rates with $CP$ session control algorithm, for changes in the average flow size (Load3, $AFS1=10000$, $AFS2=30000$)](image-url)
Figure 4.24. Periodic spikes in the average flow size

The pulses start at $t = 50$ s. The pulses vary the average flow size from 10000 to 30000 bytes. We vary the length of the pulse as well as the time interval between consecutive pulses.

4.8.1 Flow completion rate

We study the number of flows completing under varying pulse lengths and interpulse durations. We also experiment with two different flow arrival rates - 1000 flows/s (session overload), and 100 flows/s (session underload for average file size of 10000 bytes, overload for 30000 bytes). The experiments with an input flow arrival rate of 1000 flows/s are run for 200 s and those with a flow arrival rate of 100 flows/s, for 500 seconds.

4.8.1.1 High flow arrival rate

The flow arrival rate in this experiment is 1000 flows/s, and length of the pulse is 2 seconds. Figure 4.25 compares the number of flows completing in 200 s, for different interpulse durations. The number of flows completing is much higher with session control than without it. As was explained earlier, the number of flows completing with no session control increases with increase in interpulse duration,
because of the decrease in the total time under session overload. However, the increase is not as much as in Figure 4.15 because the system is always under session overload due to the high flow arrival rate. Even with session control, the number of flows completing increases with increase in interpulse duration. This is because, both session control algorithms require some time to stabilize after a change of traffic pattern. When the interpulse duration is low, there is not enough time to stabilize, resulting in oscillations. Oscillations are reduced in $CP$ due to the smoothening of the flow completion rate with history - however, that still is not enough to completely stop the oscillations.

4.8.1.2 Low flow arrival rate

The flow arrival rate in this experiment is 100 flows/s. Figure 4.26 compares the number of flows completing in 500 s, for different interpulse durations. The top plot corresponds to a pulse length of 10 seconds and the bottom plot, 2 seconds.
Figure 4.26. Comparison of the number of flows completing, with periodic pulsations in the average flows size. Top plot corresponds to a pulse length of 10 seconds and the bottom plot corresponds to pulse length of 2 seconds. Flow arrival rate is 100 flows/s
Since 100 flows/s corresponds to session underload, there is seems to be not much of an improvement using session control. In fact, using no session control completes more flows when the pulse length is 2 seconds, even though the difference is not substantial. For pulses of length 10 seconds, there is a slight advantage in using session control ($CP$) because of the increase in the session overload. Once again, the difference is not substantial.

### 4.8.2 Per-flow performance

We plot the per-flow performance (average flow completion time) in Figure 4.27. The flow arrival rate is 100 flows/s.

Results with flow arrival rate of 1000 flows/s are not shown since there is an obvious improvement due to the inherent session overload. On the other hand, 100 flows/s corresponds to session overload for a flow size of 30000 bytes only (and not for 10000 bytes). Hence it is a more interesting case as the system moves from session underload to overload. The top plot (of Figure 4.27 corresponds to a pulse length of 10 seconds, and the bottom plot corresponds to pulse length of 2 seconds. There is a huge improvement in the average flow completion times using the $CP$ session control algorithm, over no session control. Only when the interpulse duration is large (40s, 50s), is the performance without session control comparable to that of $CP$. This is because, when the interpulse duration is large, the degree of session congestion decreases to negligible levels.

### 4.8.3 Fairness between short and long-lived flows

Table 4.4 shows the average size of all flows that complete in 500 seconds. The average flow arrival rate is 100 flows/s. The average size of completed flows is highest when using $CP$. As expected, the improvement is more obvious for larger pulse lengths and smaller interpulse durations.
Figure 4.27. Comparison of the average flow completion time, with pulsations in the average flows size. Top plot corresponds to a pulse length of 10 seconds and the bottom plot corresponds to pulse length of 2 seconds. Flow arrival rate is 100 flows/s.
Table 4.4. Average size of completed flows with pulsations in input flow size

<table>
<thead>
<tr>
<th>Pulse Length (s)</th>
<th>Interpulse Duration (s)</th>
<th>Avg Flow Size Completed (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No SC</td>
<td>GSS+GA</td>
</tr>
<tr>
<td>2</td>
<td>11344.45</td>
<td>15798.56</td>
</tr>
<tr>
<td></td>
<td>10590.17</td>
<td>10534.34</td>
</tr>
<tr>
<td></td>
<td>10798.28</td>
<td>10136.85</td>
</tr>
<tr>
<td>5</td>
<td>12484.70</td>
<td>15579.98</td>
</tr>
<tr>
<td></td>
<td>11426.24</td>
<td>11528.51</td>
</tr>
<tr>
<td></td>
<td>10394.97</td>
<td>10537.70</td>
</tr>
<tr>
<td></td>
<td>10470.45</td>
<td>10302.89</td>
</tr>
<tr>
<td>10</td>
<td>13382.01</td>
<td>13809.85</td>
</tr>
<tr>
<td></td>
<td>12282.27</td>
<td>15197.52</td>
</tr>
<tr>
<td></td>
<td>10671.65</td>
<td>11861.78</td>
</tr>
<tr>
<td></td>
<td>10398.41</td>
<td>10654.57</td>
</tr>
</tbody>
</table>

4.9 Change in bottleneck capacity

The optimal admit rate for a given traffic mix would also depend on the available capacity at the bottleneck link. This is because, the available bandwidth determines how many simultaneous flows can share the link and still maintain a high flow completion rate and low flow completion time. Higher the background traffic, lower the available bandwidth, lower the maximum number of simultaneous flows possible and smaller is the optimal admit rate. To observe the effect background traffic has on the optimal admit rate (and the convergence of our algorithms), we experiment with changes (both increase and decrease) in the bottleneck capacity during the simulation run. Once again, we vary the rate at which the changes occur. Figure 4.28 shows the various load scenarios considered. Note that $BW_1$ can also be greater than $BW_2$ for increases in bottleneck capacity. Since, in reality, background traffic takes up precious router buffer space also, in our experiments, we change the buffer capacity along with changing the bottleneck link capacity.
Figure 4.28. Different load change scenarios considered for changes in the available bottleneck bandwidth - Load1 (top) to Load2 (bottom)

4.9.1 Increase in the available bandwidth

We increase the capacity of the bottleneck link from 10 Mbps to 20 Mbps, in this set of experiments. The average flow size is fixed at 10000 bytes (Pareto distributed).

4.9.1.1 Flow completion rate

We experiment with two different flow arrival rates - 1000 flows/s (session overload for both 10 Mbps and 20 Mbps) and 100 flows/s (session underload for both 10 and 20 Mbps).
Figure 4.29 shows the effect of increase in the available bandwidth during session overload. Once again, more flows complete with session control than without it. Furthermore, $CP$ performs better than $GSS + GA$ yet again.

Figure 4.30 shows the effect of increase in the available capacity when the flow arrival rate corresponds to session underload. Since increasing the available capacity makes it possible for even more flows to complete, it puts the system in an even greater underload. Hence, the number of flows completed is the same with or without session control.

4.9.1.2 Per-flow performance

Figure 4.31 shows the per-flow performance (average flow completion times) of the algorithms, under differing load conditions. Since a flow arrival rate of 100 flows/s just puts the system under more underload, we show the results only for the more interesting case of a flow arrival rate of 1000 flows/s. There is up to a factor of 3 improvement in the per-flow performance when session control is employed.

![Figure 4.29](image.png)

**Figure 4.29.** Comparison of the number of flows completing when the available bottleneck bandwidth increases from 10 Mbps to 20 Mbps. Flow Arrival Rate is 1000 flows/s
Figure 4.30. Comparison of the number of flows completing when the available bottleneck bandwidth increases from 10 Mbps to 20 Mbps. Flow Arrival Rate is 100 flows/s

Figure 4.31. Comparison of the average flow completion times, when the available bottleneck bandwidth increases from 10 to 20 Mbps. Flow Arrival Rate is 1000 flows/s
4.9.1.3 Fairness between short and long-lived flows

Figure 4.32 compares the average size of completed flows. We see that, once again, there is almost a factor of two improvement in the average flow size indicating that with session control, more large flows also complete along with smaller flows.

4.9.2 Decrease in the available bandwidth

Now, we decrease the capacity of the bottleneck link from 10 Mbps to 5 Mbps. The average flow size is fixed at 10000 bytes (Pareto distributed).

4.9.2.1 Flow completion rate

Once again, we experiment with two different flow arrival rates - 1000 flows/s (session overload for both 10 Mbps and 5 Mbps) and 100 flows/s (session underload for 10 Mbps and overload for 20 Mbps).

![Figure 4.32](image_url). Comparison of the average size of flows completed, with an increase in the available bottleneck bandwidth from 10 to 20 Mbps. Flow Arrival Rate is 1000 flows/s
Figure 4.33 plots the number of flows completing during a decrease in the available bottleneck capacity when the flow arrival rate is 1000 flows/s. Since the system is already in overload (due to the high flow arrival rate), session control yields good benefits. Yet again, *CP* out-performs *GSS + GA* even with respect to the number of flows completing.

The more interesting case of when a system goes from session underload to overload is represented by a flow arrival rate of 100 flows/s. Figure 4.34 shows the number of flows completing under such a situation. We can see that though the difference between no session control and session control has decreased, there is still an improvement in using *CP*.

### 4.9.2.2 Per-flow performance

Table 4.5 shows the average flow completion times for different flow arrival rates under different load changes. The case with a high flow arrival rate is trivial, and as expected, *CP* produces huge improvements in the average flow completion time.

![Figure 4.33](image)

**Figure 4.33.** Comparison of the number of flows completing, with a decrease in the available bottleneck bandwidth from 10 to 5 Mbps. Flow Arrival Rate is 1000 flows/s
Figure 4.34. Comparison of the number of flows completing, with a decrease in the available bottleneck bandwidth from 10 to 5 Mbps. Flow Arrival Rate is 100 flows/s

Table 4.5. Average flow completion times with decrease in bandwidth

<table>
<thead>
<tr>
<th>Flow Arrival Rate (flows/s)</th>
<th>Load type</th>
<th>Avg Flow Completion Time (s)</th>
<th>No SC</th>
<th>GSS + GA</th>
<th>CP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Load1</td>
<td>48.70</td>
<td>46.30</td>
<td>19.42</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Load2</td>
<td>47.79</td>
<td>52.91</td>
<td>19.5</td>
<td></td>
</tr>
<tr>
<td>100</td>
<td>Load1</td>
<td>5.64</td>
<td>5.81</td>
<td>6.15</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Load2</td>
<td>6.98</td>
<td>5.97</td>
<td>6.17</td>
<td></td>
</tr>
</tbody>
</table>

However, when the flow arrival rate is low (100 flows/s), the per-flow performance is approximately the same across all algorithms, across all load conditions. One of the reasons for CP not performing as well, is because of the 'initial effect' (explained in Section 4.5.4) where CP takes a longer time to reach its stable flow completion rate until which it admits less flows even in the case of underload.

However, CP does well to cope up with the decrease in bandwidth and admits only at the new completion rate. For Load2, if we consider only the flows that
complete in the last 100 seconds (i.e., after the available bandwidth has reduced to 5 Mbps), the average completion times with no session control is 22.6 s, while it is a mere 14.24 s when employing $CP$.

4.9.2.3 Fairness between short and long-lived flows

Table 4.6 shows the average size of all flows completed for different flow arrival rates for different load changes. For a high flow arrival rate, the benefit of session control is more obvious than for low flow arrival rates.

4.10 Periodic spikes in bottleneck bandwidth

As mentioned in the previous section, changes in the available bottleneck capacity can occur due to increase (or decrease) in the background traffic. Among all parameters that affect performance (input flow arrival rate, flow size and distribution, and available capacity), bottleneck bandwidth (background traffic) is the most likely to vary with time. Like other traffic and network parameters, we experiment with pulse-like changes in the bottleneck bandwidth.

Figure 4.35 shows the pulse load pattern that we subject our algorithms to. The pulse wave starts at $t = 50$ s, and continues till the end of the simulation. The length and the time interval of the pulse are varied. Two flow arrival rates are experimented with - 100 and 1000 flows/s. As discussed earlier, 1000 flows/s already corresponds to session overload, and hence is the less interesting of the two in observing the effect of the pulsations. Experiments with flow arrival rates of 1000

<table>
<thead>
<tr>
<th>Flow Arrival Rate (flows/s)</th>
<th>Load type</th>
<th>Avg Flow Size Completed (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Load1</td>
<td>No SC</td>
</tr>
<tr>
<td>1000</td>
<td>Load1</td>
<td>4620.65</td>
</tr>
<tr>
<td></td>
<td>Load2</td>
<td>4651.58</td>
</tr>
<tr>
<td>100</td>
<td>Load1</td>
<td>8512.87</td>
</tr>
<tr>
<td></td>
<td>Load2</td>
<td>8500.69</td>
</tr>
</tbody>
</table>

Table 4.6. Average size of completed flows with decrease in bandwidth
flows/s are run for 200 seconds, and those with flow arrival rates of 100 flows/s are run for 500 seconds. The bandwidth is changed from 10 Mbps to 5 Mbps during each pulse.

### 4.10.1 Flow completion rate

We measure the number of flows that complete during the experiment run, for varying flow arrival rates and load changes.

#### 4.10.1.1 High flow arrival rate

Figure 4.36 compares the number of flows completing in 200 s when the flow arrival rate is 1000 flows/s. The length of the pulse wave is 2 seconds. As expected, since the system is already in overload due to the high flow arrival rate, session control has a huge impact in increasing the number of flows completing. There is up to a factor of three increase in the number of flows completing.
Figure 4.36. Comparison of the number of flows completed with pulse-like variations in the available bandwidth. Flow arrival rate is 1000 flows/s and length of pulse is 2 seconds

4.10.1.2 Low flow arrival rate

Figure 4.37 shows the more interesting case of flow arrival rate of 100 flows/s. As a result, when the bandwidth changes from 10 Mbps to 5 Mbps, the system goes from underload to overload. The top plot corresponds to a pulse length of 10 seconds, and the bottom one, 2 seconds. The interesting observation from this figure is that having no session control performs at least as good, and in most cases slightly better than both the session control algorithms when it comes to maximizing the flow completion rate. Changing the bandwidth seems to affect the flow completion rate much more than changing the file size. As a result, faster the variation in the bandwidth, the more difficult it is for the session control algorithms to adapt to these variations. Hence we observe something contrary to what we have seen so far - when the pulses are small and rapid, the session control algorithms perform worse than no session control. However, in none of the cases is the performance of the session control algorithms substantially worse than having no session control.
Figure 4.37. Comparison of the number of flows completed with pulse-like variations in the available bandwidth. Flow arrival rate is 100 flows/s. Length of pulse is 10 seconds for the top plot and 2 seconds for the bottom plot.
4.10.2 Per-flow performance

Here, we report the effect of the pulsations on the per-flow performance (average flow completion time). The flow arrival rate is 100 flows/s. The more trivial case of 1000 flows/s is not reported, since it constitutes continuous session overload and does not show the effect of periodic sharp transitions from underload to overload.

Figure 4.38 compares the average flow completion times for pulse lengths of 10 s (top plot) and 2 s (bottom plot). We observe the pattern we have been observing with similar pulsations - when the total time of overload is high (corresponding to larger pulse lengths and smaller interpulse durations), the average flow completion time obtained by employing $CP$ session control algorithm is better (up to almost a factor of two improvement). However, when the fraction of session overload becomes lesser, no session control performs even better (though only marginally), than $CP$. $GSS + GA$ is very unstable under such rapid changes, resulting in poor flow completion times.

4.10.3 Fairness between short and long-lived flows

Table 4.7 presents the average sizes of the flows that complete, under different load situations and algorithms. As can be derived from the per-flow performance results seen in the previous section, for cases with large periods of session overload, $CP$ is most fair with the average flow size being the highest. As the fraction of session overload decreases, the advantage of $CP$ also reduces as compared to no session control.

4.10.4 Time behavior of the session control algorithm

We plot (Figure 4.39) the responsiveness of the $CP$ session control algorithm for a pulse length of 2 seconds and an interpulse duration of 20 seconds. The flow arrival rate is 1000 flows/s. Note how the flow admit rate closely follows the flow completion rate during each pulse, making the algorithm very responsive to sudden oscillations in the background traffic.
Figure 4.38. Comparison of the per-flow performance (indicated by the average flow completion time) with pulsations in the bottleneck bandwidth. Pulse length is 10 seconds for the top plot and 2 seconds for the bottom plot.
Table 4.7. Average size of completed flows with pulses in bottleneck bandwidth

<table>
<thead>
<tr>
<th>Pulse Length (s)</th>
<th>Interpulse Duration (s)</th>
<th>Avg Flow Size Completed (Bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>No SC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>8003.93</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>9456.08</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>9638.99</td>
</tr>
<tr>
<td>5</td>
<td>2</td>
<td>7233.99</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>7977.72</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>9607.09</td>
</tr>
<tr>
<td></td>
<td>40</td>
<td>9867.20</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
<td>7007.12</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>7326.98</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>8812.14</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>9549.59</td>
</tr>
</tbody>
</table>

Figure 4.39. Responsiveness of the CP session control algorithm to regular pulse-like variations in the bottleneck bandwidth
4.11 Effect of long-lived flows

It is vital that any session control scheme (either threshold or search-based) be robust to a large number of long-lived flows. If the session control algorithm admits a large number of long-lived flows, all new flows will be blocked resulting in starvation. In this case, the network would become increasingly unstable and would perform much worse than it would without any session control. Since our session control algorithms rely on the flow completion rate to decide the admit rate, when the flow completion rate is close to zero, the admit rate also reduces to a very small value. We handle this issue in our algorithms by ensuring that the admit rate never goes below an accessibility threshold ($\alpha_{min}$ as described in Algorithms 1 and 2) of the network. Accessibility threshold defines how accessible a network is to new flows. Note that even though we rely on this threshold, it is quite independent of the input traffic patterns and network conditions. The choice of an appropriate $\alpha_{min}$ is a high level administrative decision that must be made by the network administrator.

In order to evaluate our scheme when large number of long-lived flows are present, we simulate very long lived connections that enter and stay in the system for a considerable period of time. The simulation run time is 200 seconds. Flows with an average flow size of 10000 bytes are generated in the first and last 30 seconds of the simulation run. In the intermediate period, flows with an average flow size of 1 GB are generated. The rest of the simulation settings are the same as the previous experiments. The value of $\alpha_{min}$ is set to 20. We experiment with two different flow arrival rates - 100 and 1000 flows/s.

4.11.1 Flow completion rate

Figure 4.40 shows the number of flows completing with and without our session control schemes, under the two flow arrival rates experimented. With a high flow arrival rate, there is a huge improvement in the number of flows completing with session control, than without it. However, under low flow arrival rates, the CP algorithm performs worse than the other two cases - no session control and GSS +
Figure 4.40. Comparison of the number of flows completing in the presence of extremely long-lived flows.

GA. GSS + GA admits flows at rates higher than $\alpha_{\text{min}}$ in an effort to increase the flow completion rate. On the other hand, since the goal of CP is to admit at rates as close as possible to the completion rate, the maximum rate at which it can admit is $\alpha_{\text{min}}$. This is because, when the long-lived flows start, the flow completion rate becomes very low - less than $\alpha_{\text{min}}$. Having no session control also completes more flows since it admits at rates (much) higher than $\alpha_{\text{min}}$, causing the flow completion rate to be higher when long-lived flows stop arriving.

Figure 4.41 shows the time variation of the CP algorithm (admit and completion rates). As we argued, the admit rate becomes equal to $\alpha_{\text{min}}$ after the flow completion rate drops to less than $\alpha_{\text{min}}$. Comparing this with Figure 4.42 (which shows the corresponding plot for no session control), we note that the flow admit rate is much higher causing an increased flow completion rate when the long-lived flows actually do stop.
Figure 4.41. Time variation of the flow admit and completion rates with $CP$ session control algorithm, during the presence of extremely long-lived flows

Figure 4.42. Time variation of flow admit and completion rates with no session control, during the presence of extremely long-lived flows
4.11.2 Per-flow performance

Figure 4.43 compares the per-flow performance (average flow completion time) during the presence of long-lived flows for different flow arrival rates. Session control greatly improves the flow completion times when the flow arrival rate is high. For smaller flow arrival rates, having no session control performs almost as good as any of the session control algorithms.

![Comparison of the average flow completion time with and without session control, during the presence of extremely long-lived flows](image)

**Figure 4.43.** Comparison of the average flow completion time with and without session control, during the presence of extremely long-lived flows.
CHAPTER 5

DEPLOYMENT ISSUES

In the previous chapters, we motivated, detailed and evaluated our session control algorithm under a variety of static and variable traffic and network conditions, and demonstrated the significant performance gains obtained. In this chapter, we discuss some important issues pertinent to deploying our scheme at the routers.

5.1 Deployment

Since congestion is usually present at the access links (and not the links in the backbone of the network that are sufficiently over-provisioned), we envision deployment of our scheme mainly at border routers that connect companies / organizations with their ISP. However, our scheme is amenable to being deployed at the core routers also. Our scheme has a lot of advantageous features making it suitable for deployment.

- **No per-flow state at routers:** Both the session control algorithms we presented, do not require storage of any per-flow information. The per-flow computation is also negligible - only a counter needs to be incremented for every new flow admitted into the system.

- **No assumptions about input traffic characteristics:** The schemes that we propose are intended for elastic TCP flows, and not for streaming flows. Although the assumption of knowing the nature of incoming traffic might hold for streaming flows, it does not necessarily hold true for elastic flows that send data at variable rates and in an unpredictable manner. Hence, any scheme that either assumes the per-flow requirement or guarantees per-flow minimum throughput for elastic flows is not apt for real world deployment.
Both our session control algorithms, unlike other related work on session control for elastic flows, do not make any assumptions about the bandwidth requirement of incoming flows.

- **Always tend towards optimal performance:** Both our session control algorithms search for the optimal admit rate in order to optimize either of two goals - maximize the flow completion rate or maximize the per-flow performance. Since they do not use any fixed thresholds in making their admit decisions, they always tend towards optimal performance given any network setting and traffic pattern.

- **Robust to changes in traffic patterns:** Both our session control algorithms are search-based. They do use any set thresholds in making the admission control decision. When there is a change in the input traffic pattern or the network setting, they automatically converge to the new optimum value, without requiring the network administrator to modify any thresholds. Hence, they are very robust to traffic pattern changes making it very amenable for deployment.

- **No end host TCP modifications:** Since our session control solutions are entirely based at the bottleneck router, there needs to be no change at the end host TCP. Both our schemes rely on the usual TCP back-off and retransmit mechanism, to retransmit the SYN packet.

- **Simple and fast algorithms:** Both our session control algorithms are extremely simple to implement at the router. Furthermore, the admit rate decision is made every epoch,\(^1\) and not for every new incoming flow. Hence, they can occur in the router’s slow path (similar to routing updates) and do not affect the fast forwarding path. The only decision that needs to be made for every new flow is a comparison against a counter.

\(^1\)We use 2 second epochs for \(GSS + GA\) and 1 second epochs for \(CP\).
**Independent deployment**: Since the primary deployment is at the border routers, we believe that independent deployment at different routers would still yield benefits.

### 5.2 Admission control issues

Admission control of TCP flows (i.e., blocking of new sessions) can be done in many ways [20] including dropping SYN packets, or sending RST or ICMP SourceQuench packets back to the source. We use dropping of TCP SYN packets for session control since it incurs smaller amount of overhead in comparison to creating and sending RST or ICMP packets.

As we detailed in our algorithms, the session control decision in our scheme is based on measuring the flow completion rate at a router. In our scheme, routers measure the flow completion rate by tracking FIN packets traversing through them. There are a few potential hurdles with this approach. Since the Internet is connectionless at the network layer, different packets from a single connection may go through different routers. A router may see the SYN of a connection but not the corresponding FIN, or vice versa. This would skew the flow completion rate measure at the router. However since such events are rare in the current Internet [18], we believe that this will not pose a serious problem to the working of our scheme. Moreover, since we are measuring only the rate of connection departure (and not an absolute value of the current number of flows in the system), a skew incurred by an error in measuring the flow completion rate during one time interval, does not accumulate and increase with time. Furthermore, if the algorithm is running at a campus or organization gateway, this problem would not arise since all packets have to pass through it.

Since our algorithms require to snoop into the transport header of each packet to check if it is a connection initiation (SYN) or termination (FIN) packet, they would obviously skip packets whose transport headers are encrypted (such as IPSec)

---

2In scenarios where data transfer is mainly from a server to a client dropping SYN+ACK might be more appropriate.
packets) or encapsulated (such as IP in IP). Currently, less than 0.15% of all packets are IPSec packets [17]; hence this is not a very big hurdle for our scheme. However, it might become more crucial in the future with increase in IPSec traffic.

5.3 UDP flows

In this work, we focus on session control for elastic TCP flows only and not for UDP flows. Since UDP flows are not bound by any packet level congestion control mechanisms, we believe that controlling congestion at the session-level alone would not yield any benefit. Moreover, since UDP flows do not necessarily have identifiable connection setup and teardown messages, keeping track of the arrival and departure of UDP flows will involve maintaining (and updating) per-flow state at the router. This is clearly undesirable. However, we do account for the impact of UDP traffic on the network, when dealing with TCP session control. The presence of UDP flows manifests itself as a reduction in the bottleneck link capacity and an increase in router buffer occupancy. We showed in Chapter 4 how our algorithm adapts to both increasing and decreasing bottleneck capacity.
CHAPTER 6

OVERVIEW OF RELATED WORK

Call admission control has been developed and used extensively in telephone [13] and ATM networks [8, 21, 26, 22, 1]. It has also developed as flow admission control in the IntServ [4, 11, 3, 9] networks. The goal is to reserve (and police) resources as specified by the incoming flows and schedule access to resources to provide statistical or hard performance guarantees. However, because of the variations in demands of Internet applications and the complexity in specifying and policing the per-flow needs, these schemes have not been widely deployed in the Internet.

Arguments for admission control for elastic traffic were first made by Massoulif and Roberts [16]. Later, Roberts et al. [10, 2] presented a measurement-based admission control mechanism for elastic traffic that estimates the “available bandwidth” in a bottleneck link and admits a new flow only if this estimate is greater than a minimum threshold which any new flow should achieve; else, all new flows are rejected till more bandwidth becomes available. They experimented with two different techniques for estimating the “available bandwidth” - one using a “phantom” TCP connection over the bottleneck link and measuring the bandwidth it receives; the second, measuring the loss probability over the bottleneck link and relating it to the TCP throughput.

Based on the work of estimating QoS parameters of traffic streams in ATM networks using theory of large deviations [5, 7], Mortier et al. proposed a measurement-based admission control scheme in [20] for elastic traffic in the Internet. This is a threshold-based scheme - the effective bandwidth of the traffic mix is estimated using the entropy of the input traffic and a new flow is admitted only if the packet drop rate does not exceed a certain threshold. Kumar et al. proposed
a TCP admission control scheme [15] that uses a link occupancy threshold as the admission control criterion. In this scheme, new flows are admitted only if the link occupancy is below a set threshold.

All the above schemes make assumptions about the nature of incoming traffic in making their admission decision, or use loss thresholds. Although the assumption of knowing the nature of incoming traffic might hold for streaming flows, it does not necessarily hold for elastic flows that send data at variable rates and in an unpredictable manner. Moreover, all of the above schemes are threshold based and hence suffer from the drawback of not being robust against changing network and system conditions.

Our work differs from the previous works in following significant ways. First, instead of using load thresholds, our scheme uses a search-based approach to find the optimal admit rate making the scheme more robust to changing traffic patterns. Second, it does not make any assumptions about the bandwidth requirements of network flows. Third, in addition to an increase in per-flow performance, we show significant improvement in the fairness between short and long lived flows. Last, our scheme does not starve new flows even if existing flows stay in the system for a long time.
CHAPTER 7

CONCLUSION

Packet-level congestion control schemes present in the Internet today are not sufficient for controlling congestion occurring at the flow (or session) level. Controlling congestion at the session level is vital since TCP is not stable and fair when a large number of flows interact. All previous work to control session congestion for elastic flows have important shortcomings, such as making assumptions about the nature of incoming traffic and using static thresholds in making their admission decision.

In this thesis, we have introduced two novel search-based session-level congestion control mechanisms to complement the packet level mechanisms already present. The first algorithm namely, $GSS+GA$, aims to find the admit rate that maximizes the flow completion rate. We showed that, when the input flow sizes are Pareto distributed, maximizing flow completion rate does not translate to maximizing per-flow performance. Hence, we have developed a second algorithm, $CP$ (for Close and Probe), that aims to find the maximal flow admit rate that does not exceed the flow completion rate. This, we found, maximized the per-flow performance and fairness between short and long-lived flows in the system, at the same time yielding high flow completion rates.

We have evaluated both our session control algorithms under a variety of traffic scenarios and load changes, and found that during session congestion, employing our session control algorithms has huge benefits over no session control. Comparing both our algorithms, we find that overall, $CP$ performs better than $GSS+GA$, not only in terms of optimizing per-flow performance, but also in terms of maximizing the flow completion rate. Furthermore, it is fairly robust to all changes in traffic.
patterns including pulse-like variations. Hence, in this thesis, we recommend $CP$ as a viable session control mechanism for the Internet.

Some possible directions of future work include

- **Study of session congestion timescales in the Internet:** Surprisingly, there have been no study of the prevalence and timescales of session congestion in the Internet. As a result, high response times and low throughputs have come to be accepted as the norm, even if there is huge scope for improvement. With the help of real data obtained from a functioning router, we could study the prevalence and timescales of session congestion. Not only would this serve as a motivation for deployment of session control mechanisms, but it might also help us optimize our algorithms further.

- **Deployment of our session control on real routers:** Though we have evaluated our algorithm under fairly wide traffic patterns, real deployment would yield important insights into the working of the algorithms.

- **Session congestion aware end hosts:** In the current Internet, TCP SYN and DATA packets are treated alike by the routers. Hence, the loss of a SYN packet could be either due to packet or session-level congestion. Hence the end host TCP cannot differentiate between packet and session congestion and react differently. Future work involves developing strategies of informing the end hosts about the level of session-congestion in the network.

- **Study of changes to TCP SYN back-off mechanisms:** Currently, TCP reacts differently to the loss of a SYN and a DATA packet. The loss of a SYN can be detected only with a time-out, whereas the loss of a data packet can be (and mostly is) inferred with the help of the triple duplicate mechanism. This results in lesser wait times before a data packet is retransmitted. Even if a data packet incurs a time-out, the time-out value is generally much lesser $^{1}$ than for a SYN packet. As a result of these imbalances, the loss of a SYN packet

---

$^{1}$The minimum RTO is 1 second, while the initial RTO used for SYN packets is 3 seconds. [23]
packet hurts the throughput of the connection much more than the loss of a
data packet. Hence, a possible future direction of research is to study more
aggressive TCP SYN retransmission mechanisms and the effect they have
with session control.
REFERENCES


