TCP/AQM Congestion Control Based on the H_2/H_∞ Theory

Navin Haghighizadeh

Thesis submitted to the
Faculty of Graduated and Postdoctoral Studies
in partial fulfillment of the requirements
for the Doctorate in Philosophy degree in Electrical and Computer Engineering

School of Electrical Engineering and Computer Science
Faculty of Engineering
University of Ottawa

Abstract

This thesis uses a modern control approach to address the Internet traffic control issues in the Transport Layer. Through literature review, we are interested in using the H_2/H_∞ formulation to obtain the good transient performance of an H_2 controller and the good robust property from an H_∞ controller while avoiding their deficiencies. The H_2/H_∞ controller is designed by formulating an optimization problem using the H_2-norm and the H_∞-norm of the system, which can be solved by an LMI approach using MATLAB.

Our design starts with the modeling of a router and the control system by augmenting the network plant function with the Sensitivity function *S*, the Complementary Sensitivity function *T* and the Input Sensitivity function *U*. These sensitivity functions along with their weight functions are used to monitor the closed-loop dynamics of the traffic control. By choosing different combinations of the sensitivity functions, we can obtain the SU, the ST and the STU controllers. Both the window-based and rate-based version of these different types of H_2/H_∞ controllers have been designed and investigated. We have also proved that these controllers are stable using Lyapunov's First Method.

Next, we verify the performance of the controllers by OPNET simulation using different performance measures of queue length, throughput, queueing delay, packet loss rate and goodput. Our performance evaluation via simulation has demonstrated the robustness and the better transient response such as the rise/fall time and the peak queue value. We have also investigated the controller performances subject to network dynamics as well as through comparison with other controllers.

Finally, we have improved these controllers for real-time application. They are capable to update/renew the controller in a short time whenever new network parameter values are detected so that the optimum performance can be maintained.

Acknowledgments

I would like to extend my greatest gratitude to Prof. Oliver Yang, my thesis supervisor, for his invaluable guidance and persistent support throughout my Ph.D. career at the University of Ottawa. His great intuition, profound knowledge, remarkable research expertise and inspiring comments made the work challenging while hopeful and enabled me to succeed in my study.

Many thanks go to the members of the CCNR (Computer Communication Network Research) laboratory for their help and unforgettably nice time being together.

I owe a great debt of thanks to family especially my parents, Dina and Nasser, for encouraging me to peruse my studies. I thank them for their endless love and support.

A special gratitude goes to my husband, Sobhan, for his unfailing love and constant support. I thank him for bringing peace and comfort to my life.

Finally, I gratefully acknowledge the financial support from the NSERC Research Discovery Grant (#RGPIN42878).

Table of Contents

List of Figures

List of Tables

Table of Symbols and Notations

		Page of 1st Appearance
$\boldsymbol{A, B, C, D}$	State-Space representation matrices	40
$\boldsymbol{A_{cl}, B_{cl}, C_{cl}, D_{cl}}$	State-Space representation matrices of Closed-loop Plant	42
$\boldsymbol{A_K, B_K, C_K, D_K}$	State-Space representation matrices of the Controller	42
$A_{K_{ij}}$	(*i,j*) entry of the matrix $\boldsymbol{A_K}$	43
$\boldsymbol{A_P, B_P, C_P, D_P}$	State-Space representation matrices of the Plant model	41
$A_{W_T}, B_{W_T}, C_{W_T}, D_{W_T}$	Associated values with state-space representation of W_T	54
$A_{W_U}, B_{W_U}, C_{W_U}, D_{W_U}$	Associated values with state-space representation of W_u	40
$A_{W_s}, B_{W_s}, C_{W_s}, D_{W_s}$	Associated values with state-space representation of W_s	40
$\boldsymbol{A_x}$	State matrix (Generalized plant)	19
a_1, a_2, a_3	W_T Parameters	22
B_{K_i}	i^{th} element of column vector $\boldsymbol{B_K}$	43
$\boldsymbol{B_1}, \boldsymbol{B_2}$	Coefficient matrices for $\boldsymbol{x}$ (Generalized plant)	19
C	Link capacity	5
C_{K_i}	i^{th} element of row vector $\boldsymbol{C_K}$	43
$\boldsymbol{C_{cl1}, D_{cl1}}$	Coefficient matrices for output $\boldsymbol{Z_\infty}$ (Closed-loop plant)	42
$\boldsymbol{C_{cl2}, D_{cl2}}$	Coefficient matrices for output $\boldsymbol{Z_2}$ (Closed-loop plant)	42
$\boldsymbol{C_y, D_{y1}, D_{y2}}$	Coefficient matrices for output $\boldsymbol{y}$ (Generalized plant)	19
$\boldsymbol{C_2, D_{21}, D_{22}}$	Coefficient matrices for output $\boldsymbol{Z_2}$ (Generalized plant)	19
$\boldsymbol{C_\infty, D_{\infty 1}, D_{\infty 2}}$	Coefficient matrices for output $\boldsymbol{Z_\infty}$ (Generalized plant)	19
$e(\infty)$	Steady state error	32
$G(s)$	Generalized plant	15
I	Identity Matrix	14
$K(s)$	Controller	14
$L(s)$	General open loop transfer function	22
N	Number of TCP sessions	5
N_{loop}	Loop iteration values	113
$P(s)$	Plant Model	13
p_0	Drop packet probability equilibrium point	28
$P_{queue}(s)$	Queue model transfer function	27
$P_{tcp}(s)$	TCP plant transfer function	27
$q(t)$	Instantaneous queue length	81
q_{ref}	Reference Queue length	82
q_0	Queue length	82
R_0	Round Trip Time	5
$S(s)$	Sensitivity function	14
$T(s)$	Complementary Sensitivity function	14
T_{qu},	Transfer function between $q(t)$ and $u_i'(t)$	82

t_{temp}	real-time controller's checking time interval	113
T_r	Rise/fall time	32
$\boldsymbol{u}$	Control signal/input vector	14
$U(s)$	Input sensitivity function	14
$u_i\ (t)$	Current sending rate of source *i*	81
$u_i'(t)$	Allowed sending rate of source *i*	81
U_0	Transmission rate operating points	82
$\boldsymbol{w}$	Uncontrolled traffic	15
$W_S(s)$	Sensitivity weight transfer function	14
$W_T(s)$	Complementary weight function	14
$W_U(s)$	Input sensitivity weight transfer function	14
$win_i'(t)$	Advertised window size	82
W_0	Advertised window size operating points	82
$\boldsymbol{x}$	State vector	40
$\boldsymbol{X, X_2, X_\infty, X_{pol}, Q}$	Lyapunov Matrices	19
$\boldsymbol{x_{cl}}$	Closed-loop state vector	42
$\boldsymbol{x_p}$	Plant state vector	41
$\boldsymbol{Y}$	Output Matrix	19
$\boldsymbol{y}$	Output vector/plant output	14
y_{ref}	Reference Queue Length/Input	14
$\boldsymbol{z}$	Output vector	15
$\boldsymbol{Z_2}$	Associated output with the H_2 performance	15
$\boldsymbol{Z_\infty}$	Associated output with the H_∞ performance	15
α	Associated weights with $\|\boldsymbol{E}_\infty\|_\infty$	18
β	Associated weights with $\|\boldsymbol{E_2}\|_2$	18
δ_0	Associated upper-bounds with $\|\boldsymbol{E}_\infty\|_\infty$	18
$\boldsymbol{\delta}$	Closed-loop H_∞-norm	19
δ_p	Packet drop probability deviation	27
δ_q	Queue length deviation	27
γ	H_2/H_∞ norm	18
μ_{temp}	Mean queue length in a time interval of t_{temp}	113
$\mu_{th\text{-}temp}$	Mean queue length temporary threshold	113
μ_{Final}	Mean queue length final threshold	113
μ_q	Mean queue length	32
$v(t)$	Incoming aggregate uncontrolled flow rate	81
ν_0	Associated upper-bounds with $\|\boldsymbol{E_2}\|_2$	18
σ_{temp}	Queue length standard deviation in a time interval of t_{temp}	113
$\sigma_{th\text{-}temp}$	Queue length standard deviation temporary threshold	113
σ_{Final}	Queue length standard deviation final threshold	113
σ_q	Standard deviation of the queue length	32
τ_{bi}	Feedback delay from the destination *i* back to the source *i*	13
τ_{ei}	Time delay of a packet from source *i* to the router	13

τ_{fi}	Time delay from the router to the destination *i*	13
τ_i	Time delay for source-destination pair *i, RTT*	13
ω	Frequency	16
ω_B, *M*, *A*	W_s Parameters	21
$\|\boldsymbol{E}_\infty\|_\infty$	Closed-loop H_∞-norm	16
$\|\boldsymbol{E_2}\|_\mathbf{2}$	Closed-loop H_2–norm	16
$\left(\begin{array}{c\|c} A & B \\ \hline C & D \end{array}\right)$	Shorthand for the state-space realization $\boldsymbol{C}(s\boldsymbol{I}-\boldsymbol{A})^{-1}\boldsymbol{B}+\boldsymbol{D}$	40

Table of Acronyms and Abbreviations

		Page of 1st Appearance
ACK	Acknowledgement	12
API-RCP	Adaptive Proportional-Integral Rate Control Protocol	3
AQM	Active Queue Management	2
BDP	Bandwidth-Delay Product	2
cwnd	Congestion Window Size	75
ECN	Explicit Congestion Notification	2
ECT	ECN-Capable Transport	3
FIFO	First In First Out	17
GM	Gain margin	32
HSTCP	High Speed TCP	2
LMI	Linear Matrix Inequality	5
MIMO	Multiple Input Multiple Output	4
PI	Proportional Integration	4
PM	Phase Margin	32
QCS	Queue Control Scheme	5
QoS	Quality of Service	1
RB-STU	Rate-Based STU	9
RB-SU	Rate-Based SU	9
RCP	Rate Control Protocol	3
RED	Random Early Detection	2
RT	Real Time	113
RTPD	Round Trip Propagation Delay	73
RTT	Round Trip Time	3
SISO	Single Input Single Output	4
ssthresh	Slow Start Threshold	149
ST	Sensitivity/Complementary sensitivity Controller	9
STCP	Scalable TCP	2
STU	Sensitivity/Complementary sensitivity/Input sensitivity Controller	9
SU	Sensitivity/Input sensitivity Controller	9
TCP	Transmission Control Protocol	1
VoIP	Voice over IP layer	1
XCP	eXplicit Control Protocol	3

Chapter 1
Introduction

The QoS (Quality of Service) of a data network would deteriorate due to network congestion when a link carries more data than its capacity. When setting up a new connection, high packet loss rate, long queueing delay and inability are typical effects of congestion in a network. Modern networks often use congestion reactive control and congestion avoidance techniques to prevent network from collapsing. However, they still suffer from various problems. Applications such as VoIP (Voice over IP layer), multimedia communication with contents such as video and audio, and online multiplayer games are all generating more data traffic than before when the traffic was mostly texts and images. For a designed network capacity, congestion will set in often when the increasing traffic is getting close to the capacity limit. It is desirable to design a network that can provide a high data rate but without congestion in order to improve the video and audio service while simultaneously guaranteeing efficient web and email transmission. A good congestion controller would be able to handle/alleviate all the undesirable symptoms/performance (long queue length, delay, high loss rate, …) brought about by congestion. In other words, congestion controllers continue to be an interesting research topic as network types (and their traffic) evolve. In this chapter, we shall first go through the historical development of congestion control protocols with the pros and cons of each, and then based on which, we propose our motivations, objectives and methodology that would serve to designing a new congestion controller.

1.1 Some Historical Development of Congestion Control

The first designed Internet network was based on best effort service model [GeCr01] that was a completely uncontrolled network and might suffer from network collapse due to congestion. Congestion control algorithms can be classified according to different criteria. Implicit versus explicit signaling method is one such classification.

In an implicit signaling method, congestion is construed from different information fed back

from the destination or path entities. For example, the traditional TCP (Transmission Control Protocol) [Jaco88] treats the network as a black box [RaFl99], i.e. it has no network information such as link bandwidth and network traffic loads. A source determines congestion has occurred when a packet is lost and adjusts its window size accordingly. This can cause a periodic oscillation in the window size and possibly induce chaotic behavior into the network, thus adversely affecting overall network performance such as utilization, fairness and stability [FeVa03, HaBe07].

TCP variants have been proposed to overcome the shortcomings of TCP. For examples, FAST TCP [WeJi07] uses queueing delay as the primary measurement of congestion to adjust the congestion window. Although this scheme can reduce more packet-level oscillations [ChWe05], it may also artificially introduce queueing at the bottleneck router [QaZn09]. HSTCP (High Speed TCP) [Floy03] proposes a large congestion window to improve the flow throughput [Floy03], but it is still packet-loss based. TCP Vegas [BrMa94] estimates the available bandwidth to improve the utilization and to reduce the packet loss, but investigation [HeBo00] found that its congestion avoidance mechanism has fairness problems even if all competing connections operate with the same round trip time. The STCP (Scalable TCP) protocol [TeSz05] improves the link utilization in high BDP (Bandwidth-Delay Product) networks but unfriendly pushes out other existing regular TCP flows [TeSz05]. In summary, although these TCP variants indeed show the improved performance to TCP Reno in some aspects, they still suffer some potential problems as discussed above. Furthermore, since all these protocols are still implicit, they lack the accurate knowledge of traffic levels in the network, and therefore have to behave conservatively. An application may have to attempt a few times before it can meet the available bandwidth limit [PoDh06].

Other improvements include the more efficient AQM (Active Queue Management) algorithms that can overcome the defect of Drop-Tail mechanism in a router. However, all these implicit protocols have to behave conservatively due to lack of precise network information. For example, RED (Random Early Detection) algorithm [FlJa93] probabilistically discards/marks the packets by observing the average queue size so that the queue always operates in a reasonable size [FlJa93]. Instead of dropping/marking packets, ECN (Explicit

Congestion Notification) [RaFl99] explicitly feeds back congestion signal with ECT (ECN-Capable Transport) code-points to sources. In these ways, a TCP source can actively adjust its sending behavior before the queue overflows. The advantage of both RED and ECN methods is that they only use the queue size to decide if a packet should be dropped or marked in that they do not need to estimate link states such as bandwidth capacity. Nevertheless, they have some other problems, For examples, TCP+RED [LoPa02a] inevitably causes severe oscillations in the source throughput as the link bandwidth or latency increases, while TCP+ECN [LeMo01] still inherits the bias against connections with long RTTs (Round Trip Times) as well as the unfairness towards new connections. However, the sources in these two protocols still have to behave conservatively because their feedback is still highly imprecise [QaZn09]. Besides, as shown in [HoYa07, HoYa10], TCP+RED exists great fluctuations in both the router queue size and flow throughput when drastic network changes happen.

In light of the above issues with AQM and TCP, explicit congestion control protocols have been used to signal the network congestion level more explicitly. Examples are the XCP (eXplicit Control Protocol) [KaHa02], RCP (Rate Control Protocol) [DuMc06], API-RCP (Adaptive Proportional-Integral Rate Control Protocol) [HoYa07], JetMax [Zhle06], MaxNet [WyAn03] and utility function-based method (e.g., [KeMa98, LoLa99]). They record link information in a dedicated field of packet header and feed it back to the source so that the bandwidth could be efficiently utilized. Specifically, JetMax, MaxNet and utility function-based method signal the required fair rate or link price back to the source, but the final sending rate is decided by the source itself. XCP feedbacks the required increment or decrement of the sending rate, while RCP and API-RCP directly signal the admissible sending rate to the source. The advantages of these protocols are that they provide the explicit information to signal the dynamic traffic levels in a link without keeping per-flow state. In addition, the fully explicit congestion control may allow the design of a fair, stable, low loss, low delay and high utilization network [KeRa10].

Despite the advent of the explicit congestion control protocols and its many investigations done in past ten years, due to the complexity and the difficulty in implementation, no explicit controller has been practically deployed. As such, the networks nowadays still heavily rely on the implicit AQM congestion controllers. However, in the presence of network plant uncertainty

like link capacity, RTT and load, AQM can potentially behave badly. Experimental evidences show that RED has no robust performances under uncertain network conditions like traffic changes [HoYa06]. Several mathematical models of TCP dynamics e.g., [Masc99, MiGo00 and LoPa02a] have been proposed in order to use control theory to exercise congestion control. For instance, an improved approach using PI (Proportional Integration) controller has been proposed in [HoMi02] for tuning the RED parameters [MiGo00] to overcome buffer overflows, packet loss and link under-utilization problems. Many similar studies have been done like P (Proportional) controllers [HoCh02a, ChYa03a, ChYa03b], other different PI controllers [HoCh02b, Chen04], Phase margin based controllers [HoYa04], pole placement based controllers [ChYa04a, ChYa04b, ChYa04c]. However, the systems using these controllers respond sluggishly in the present of large plant uncertainty [QuOz04].

The controllers mentioned above are well-known classical controllers. Due to historical development, there are limitations to the "classical" controllers that rely mostly on linear classical control theory. In contrast, more recent "non-classical" work has appeared for controllers that can be linear or nonlinear, time-invariant or time-variant [Ogat02] and can be applicable to both SISO (Single Input Single Output) and MIMO (Multiple Input Multiple Output) systems. Examples are the optimization approaches (e.g., [LoPa02a]), the game theory (e.g. [AlBa02]), the RED-like controller design using fuzzy logic (e.g., [GuYa07]) and those designs using economic methods (e.g., [DuNi03, Stid04]). Although some modern control methodology actually has a long history such as the Lyapunov's Stability Criterion [RoHa77]. H_∞ based congestion is proposed in order to achieve a robust AQM, which will be reviewed in detail in the next section.

The Internet congestion control algorithm can also be classified into window-based and rate-based. In window-based control, a source transmits data with a congestion window-size which is changed whenever congestion is detected. Examples are the TCP and its variants discussed earlier. In rate-based control, a source transmits data by adjusting its rate according to the traffic intensity of network. While there are still many recent work that are window-based (e.g. [AbSh15]), it has also been shown that window-based controllers such as the RED and its variants are inherent unstable [LoPa02a, LoPa02b].

In general, a rate-based control algorithm often leads to a relatively smooth source transmission rate and short router queue length as well as a high network utilization and fair allocation of the network bandwidth [GeLo02]. Examples are the RCP [DuMc06], API-RCP [HoYa10] and the QCS (Queue Control Scheme) [HuXi09]. Both the RCP and the API-RCP controllers show network bandwidth fairness and smooth throughput. However, the RCP may suffer continuous oscillations due to misestimating the bottleneck link capacity, which can be severe when such capacity is time-varying [ZhAh05, ZhHe05], while the API-RCP may experience relatively large queue size oscillations because of its adaptive PI scheme, which involves switching between PI controllers [LiBe03]. The QCS [HuXi09] is based on instantaneous queue size but suffers from large queue size oscillation and queue overflow in case of drastic changes of load traffic. Some modern controllers are also rate-based. Examples are the H_∞ controller [QuRa01] and Fuzzy logic controller [PiPr06, LiYa11]. These controllers suffer from poor transient responses with relatively large rise/fall time and exclusively large oscillation around steady-state queue size for [LiYa11].

1.2 H∞-based Congestion Controllers

Since 2000, there have been some applications of H∞ theory in the network congestion control. As one important approach of the "modern control" methodology, they have shown good performance as well as having various shortcomings. An important issue on designing a robust controller is modeling different dynamic network parameters. The H∞ controllers can be applied to a system to achieve robust performance despite the network uncertainties in link bandwidth *C,* RTT R_0, and link load *N*. For instance, measuring available bandwidth of a link is difficult and complex, and thus it can be modeled as a constant value plus disturbance [ZhNe07, ZhNe09, ChKu11]. The controller introduced in [ZhNe09] has been designed by assuming that the link bandwidth is the only uncertainty in the network. It has studied H∞ controller in AQM network by designing a controller in TCP/AQM network with constant bandwidth and time variant disturbance. Also, the H∞ controller has been designed for a wireless network in [ZhNe07, ChKu11] by assuming only variant link bandwidth as disturbance. Also, LMI (Linear Matrix Inequality) approach was employed to solve the H∞ problem in time domain [ZhNe07].

Although the results in above works show more robust performance than RED, the queue size still has significant fluctuation around equilibrium point.

The H∞ controller designed for a single bottleneck network in [QuRa01] has assumed only the delay as an uncertainty in the network, which is not applicable to the real network because there is no determinant knowledge about other parameters either such as link bandwidth and traffic load. More realistic assumptions have been used in [QuOz04, ChYa05, ChYa06, ChYa07] by considering C, R_0 and N as uncertainties of the network like a constant value plus disturbances. An AQM controller has employed an H_∞ controller based on the plant uncertainty [QuOz04, MaBe07, ChYa07]. In addition, a loop-shaping PI controller has been introduced [AlHa09] to solve the mixed sensitivity problem in the design of a robust H_∞ controller. In addition, an H∞ mixed sensitivity problem using μ-optimal approach has been used to design an H∞ controller for TCP/AQM networks [ChYa07] where the nominal performance, robust stability, and robust performance were analyzed. Although the controllers in the above-mentioned papers have shown a better performance than PI and RED controllers, they have significant fluctuation around the equilibrium point of the queue size even in steady state and poor transient response on overshoot and the rise time. Finally, there are also works on H_2 [Mack04] but performance evaluation shows that H_2 controller is sensitive with respect to perturbations and robustness cannot be guaranteed.

Modern controllers can also be classified as window-based or rate-based. The SU H∞ and the STU H∞ controllers [ChYa05, ChYa07] have considered TCP/AQM plant model in frequency domain and have applied classic or modern control theory to design the controller in s-domain. The controllers control the queue size of the router by dropping the packets not randomly (like RED) but by computing the dropping ratio using the H∞ control theory. All these controllers have window-based operation in a router. While they show better performances than other kinds of classical controllers (such as [HoMi02, ChYa04axx]) designed for TCP/AQM, they do not have satisfying transient responses like large rise/fall time, large peak value queue size.

In addition to the window-based controllers above, some are rate-based such as the H_∞ controller [QuRa01]. These controllers suffer from poor transient responses with relatively large rise/fall time.

1.3 The H_2/H_∞ Controller

The H_2/H_∞ controller has integrated the H_2 controller and H_∞ controller in order to have good transient performance from H_2 controller and good robust property from H_∞ controller. There have been different approaches to solve the H_2/H_∞ problem. The Riccati approach has been studied in several articles [KhRo91, BaSh93], but it is relatively complex to solve the problem. Therefore, the LMI approach [e.g. Sche95, HaKa99, PaGa05, YaXi06, BaEm12] has been proposed to reduce complexity order of the problem.

So far, the H_2/H_∞ controllers have been mostly used in mechanical applications such as mobile robot tracking [HwHa05, XiYa05], hard disk drive controller [ZaAm08], controller for active magnetic bearing [DaSe08, JaMo09], automotive suspension model [BaEm12] and VSTOL flight control [HaKh10]. It is also used for independent drive electrical vehicle [LiPe13] and power system [BeHi06]. They all showed an improved stability, robustness, and transient response. There is also limited work related to traffic control. The work on finding the optimum RED parameters [EsSh11] still suffers from large queue size oscillations, and the mixed sensitivity controllers of H_2 and H_∞ for the TCP/AQM networks [ToEl11] is just two separate controllers without any verification of their effectiveness.

1.4 Motivation and Objectives

After reviewing and exploring existing literature/work in Sections 1.2 and 1.3, we are motivated to design the H_2/H_∞ controllers to obtain both the good transient performance from the H_2 formulation and the good robust property from H_∞ controller while avoiding their deficiencies. We believe this technique has a potential to control traffic more effectively in the Internet nowadays. We would also like to investigate the rate-based operation in addition to the window-based operation that is inherently unstable as discussed. Some nice features (such as lower source sending rate requirement and less oscillation in the router queue) of rate-based controllers would allow our controllers to be capable of providing better performance (lower delay, loss rate, etc.) when congestion sets in.

In a real world system, network parameter values would change in time. For examples, the number of sources transmitting data or the RTT of the source-destinations connections may vary in time. The available bandwidth of a bottleneck link in a wireless network can also fluctuate as a deterministic amount of bandwidth is usually not available in a connection-based network. Hence, we would like to design a real-time H_2/H_∞ controller based on the current network parameters values sot that good performance can be maintained upon changes in the system within a short amount of time.[1]

The general objective of our work is to design robust congestion controllers using the "modern control" methodology in order to achieve desirable transient response, stability and robustness. Specifically, we would like to

1) investigate and design a window based H_2/H_∞ controller using LMI approach
2) investigate and design the rate-based H_2/H_∞ controllers.
3) investigate the controllers performances subject to network dynamics
4) Improve the controllers for real-time applications to be able to adapt network parameters changes

1.5 Methodology and Approaches

We shall design our controllers to provide robustness[2] with good transient performances to exercise congestion control in communication networks. A robust system would tolerate small fluctuation from nominal network parameters so that knowing the exact value of network parameters does not matter.

Our design will start with the modeling of a router, and then apply modern control techniques to design controllers. Then we verify the performance of the controllers by simulation. We shall evaluate different performance measures like queue length, throughput,

[1] The terminology "real-time" may have different meaning in different science/engineering fields. For computer networking research/applications in this thesis, real-time means an algorithm (or device) is able to respond within the time interval of changes in the physical system. As an example from this research, if x seconds is the shortest time the network traffic is disturbed, our controller is real-time if it is able to react and bring the node system back to the original/desirable operating point in less than x seconds.

[2] Robustness of a control system means that acceptable performance can be achieved despite the presence of significant plant uncertainty within an acceptable range [Mack04]. The uncertainty can arise from model inaccuracies and changes, or from noise and disturbance signals.

queueing delay, packet loss ratio. Of the different types of simulators available, we shall use OPNET [Opne13] because it is readily available in the research community, and our CCNR Lab has existing libraries of different types of controllers implemented in OPNET before. Unlike MATLAB [Matl13] that is equation-based**,** OPNET is packet-based, and can provide closer models to the real-world networks. More specific approaches to be carried out for each type of design are provided in the following.

1.5.1 The H_2/H_∞ Congestion Controller

In order to provide H_2/H_∞ [Sche95] congestion control, we will use the TCP/AQM plant model proposed in [HoMi01], which is based on window operation. It will also give us the opportunity to compare with other similar works. We will first define sensitivity and different input sensitivity weight functions that would allow us to study the tradeoff among conflicting objectives in the controller design. There can be different combinations of sensitivity functions, which give different mixed sensitivity controllers called Sensitivity/Input sensitivity (SU), Sensitivity/Complementary sensitivity (ST), Sensitivity/Complementary sensitivity/Input sensitivity Controller (STU) controllers. Then, we shall use the H_2/H_∞-norm formulation and the bound optimization with the LMI approach to design the H_2/H_∞ controllers. Since MATLAB [Matl13] already has many built-in functions/tools for these analysis/design techniques, we shall use it to analyze and implement our controllers. We provide a design example on the selection of the weight functions and its impacts on the performance of the TCP/AQM system for each controller. We shall compare the proposed controllers to PI-controller, the SU H_∞ and the STU H_∞ controllers to see if the system performances improve.

In order to design the rate-based controller, we first derive rate-based TCP/AQM plant model. Then we apply the same procedure as the window-based controller by defining the sensitivity and different input sensitivity weight functions. Among different combinations of the sensitivity functions, which give different mixed sensitivity controllers, we design the Rate-Based SU (RB-SU) and the Rate-Based STU (RB-STU) controllers.

We analyze the controllers in s-domain (i.e., the frequency domain) to see the impulse/transient responses such as the rise/fall time and the peak value as well as the steady

state error. We shall also use the well-known Lyapunov method in time-domain to analyze the stability of the controller. This requires finding the eigen values in state-space representation of the closed-loop system using the generalized plant model. Since there is no information on how the weight functions and the original plant model are augmented to give the generalized plant model, quite a bit of effort is involved to determine how the states are interconnected in the generalized plant model and to determine each element in the state-space matrix for G.

In addition, we want to improve the speed of the algorithm for real-time applications and we apply H_2/H_∞ controller for rate-based control. Finally, for the reason stated above, we shall use the packet-based OPNET modeler [Opne13] to verify and to evaluate the performances of the new controllers. This would require us to convert the s-domain in MATLAB to the z-domain in order to run the discrete-time-based OPNET simulation.

In order to provide the real-time H_2/H_∞ controllers, either window-based or rate-based, a new controller is designed upon a network parameter changes by calling MATLAB [Matl13] form the simulator. The performance of the new controller is observed real-time to see if it is satisfying or a new controller needs to be designed.

Finally, based on our experience on design and performance evaluation of these controllers, we shall provide some guideline for future researchers to design the H_2/H_∞ controllers.

1.6 Thesis Contributions

The following are the contributions of this research work:

1) The application of the H_2/H_∞ controllers to TCP/AQM network, which uses the LMI approach. to optimize mixed H_2/H_∞-norm objective function. To our best knowledge, we are not aware of any previous work using the same approach.

2) Designing three different types of the H_2/H_∞ controllers (the SU, ST and STU controllers) by choosing different combinations of the sensitivity functions.

3) Designing and comparing two different schemes of the H_2/H_∞ controllers: the window-based and the rate-based controllers.

4) The evaluation and the comparison of the H_2/H_∞ controller to demonstrate its robustness and its better transient response in the rise/fall time and the peak queue value as well as other

performance measures in packet loss ratio, throughput and goodput.

5) Analyzing the stability of the proposed controllers using Lyapunov's First Method.

6) Designing real-time version of the H_2/H_∞ controllers.

1.7 Thesis Organization

For the remainder of this thesis, Chapter 2 provides the network modeling and assumptions for our new schemes. Chapter 3 presents the designing principles of the H_2/H_∞ congestion controller. Chapters 4 and 5 provide the design, analysis and performance evaluation of the window-based H_2/H_∞ controllers and rate-based H_2/H_∞ controllers respectively. We provide the new scheme of real-time H_2/H_∞ controller in Chapter 6. More insights about the controller design are discussed in Chapter 7. With Chapter 8, we conclude the thesis. The appendices at the end provide additional information on some theoretical background, analysis as well as the results some repetitive experimental performances of TCP/AQM networks and H_2/H_∞ controller in order to make the presentation of our thesis more concise.

1.8 Publications

The following is a list of publication that have appeared, submitted or under preparation

1) Navin Haghighizadeh and Oliver Yang, "Internet Traffic Control with H_2/H_∞ controller", Proceed. *28th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE),* pp.220-225, 3-6 May 2015.
2) Navin Haghighizadeh and, Oliver Yang, "An SU Traffic Controller for the TCP/AQM Networks", *Proceed. 2015 IEEE International. Conf. on Communications, (ICC2015)*, pp. 6110-6115, 8-12 June 2015, London, UK.
3) Navin Haghighizadeh and Oliver Yang, "Stability Analysis of an SU Controller for the TCP/AQM Network", *Proceed 6th Inter. Conf. on the Network of the Future (NOF2015),* pp. 1-3, Sept 30-Oct 2 2015, Montreal, Canada.
4) Navin Haghighizadeh and Oliver Yang, "Internet Traffic Controller based on H_2/H_∞ Formulation" under preparation for submission.
5) Navin Haghighizadeh and Oliver Yang, "Stability Analysis of the H_2/H_∞ Congestion Controller for Internet Traffic", under preparation for submission.

Chapter 2
Network Operation, Modeling and Assumptions

In this chapter, we first describe the general network model used in this thesis we are going to investigate, and then we shall introduce general operation principal of our controller. Finally, we state the few common assumptions used throughout our research.

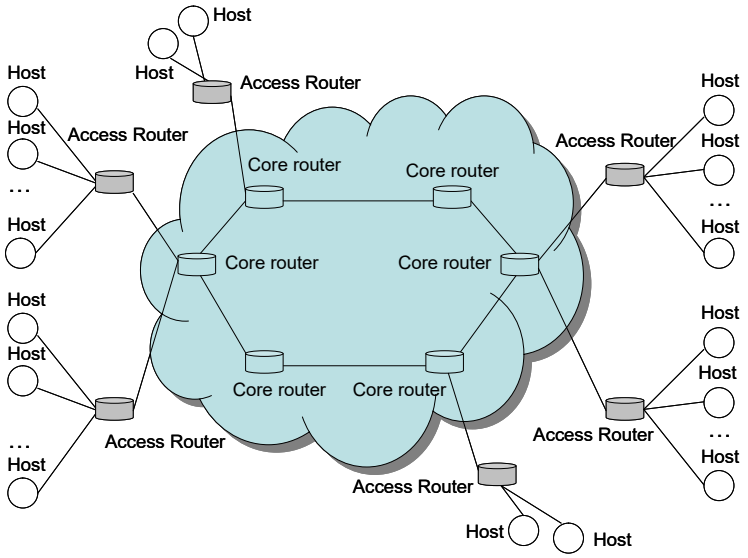


Fig. 2.1 General Network Model [TaWe10]

2.1 Network Model

We consider an end-to-end packet switching network such as the one shown in Fig. 2.1 on a general network topology. Hosts are connected to access routers which in turn are connected to core routers that form the backbone of the network. The routers forward packets from host to host to realize end-to-end communications.

The traffic generated at a source host will go to a destination host in a fixed path of access and core routers. The destination host sends the ACKnowledgment back to the source host upon receiving a packet. The time from the moment the source host sends a packet to the moment that it receives the ACK is called RTT (Round Trip Time) and designated as R_0. The link

capacity of a bottleneck link is designated by C (packets/second). Congestion can happen when the queue builds up at each bottleneck link.

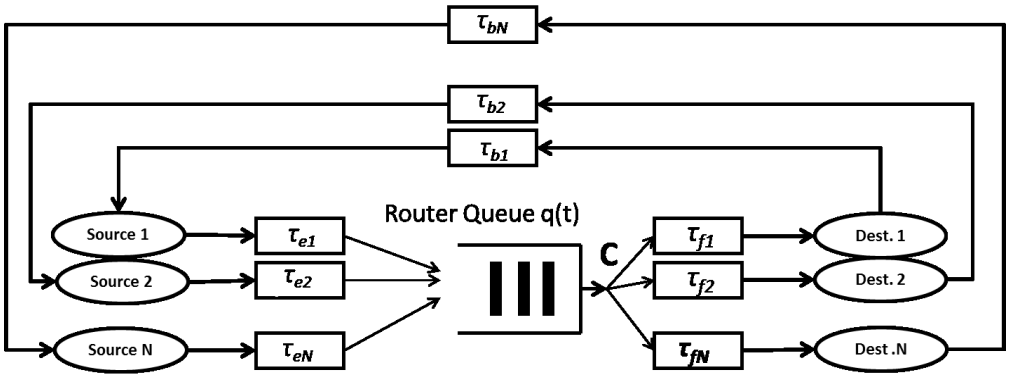


Fig. 2.2: End-to-End System Model

All the routers implement an AQM controller to control the packet rate in bottleneck links. AQM controllers take queue length variation as input and compute packet drop probability as output. The source hosts regard the packet drop as a notification of the network congestion, and act to adjust their congestion window size and hence sending rate accordingly, and in turn, this will reduce the queue size in the router. Fig. 2.2 depicts an end-to-end system model where *N* sources are sending their Internet traffic via one typical router to their respective destinations. The streams from these sources share a common buffer at the node for which we want to control the queue length at a desired level. For a particular source-destination pair *i*, τ_{ei} is the time delay of a packet from source *i* to the router, and τ_{fi} is the time delay from the router to the destination *i*, while τ_{bi} is the feedback delay from the destination *i* back to the source *i*. Obviously, $\tau_i = \tau_{ei} + \tau_{fi} + \tau_{bi}$ is the RTT.

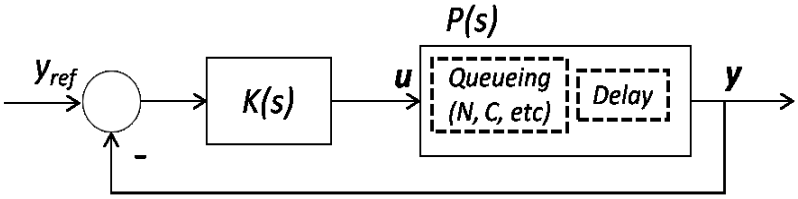


Fig 2.3: TCP/AQM Feedback Control

2.2 Control System Model for the TCP/AQM Networks

The objective of our controller is to control the queue length at the congested node (see Fig. 2.2) at a given desired level y_{ref}. The AQM system in Fig. 2.2 can be modeled as a plant model P(s) in Fig. 2.3 by including the queueing at the shared buffer and the RTT of all flows. Fig. 2.3 depicts the TCP/AQM control system in its frequency domain for which we want to design, apply and test our controller. The plant model *P(s*) in the big box models the TCP operations [HoMi01]. It is a linearized model in terms of different network parameters such as link capacity *C*, load *N*, and R_0. The values of these parameters can be measured/estimated. For examples, *N* can be estimated using the "Zombie list" [OtLo99]; the link bandwidth *C* can be measured by the service rate of the local congested router [ZhHo03]. The parameter R_0 can be estimated by using method proposed by [HaFl03] which is usually the average round trip time of all flows converging at the shared buffer of Fig. 2.2.

The box *K(s)* is the controller one wants to design to maintain the router queue length at a desired level q_{ref}. This is also referred to as the reference queue length input y_{ref} in Fig. 2.3. The output *y* is the controlled variable according to the objective of the controller. For examples, y can be the often-used queue length variation in window-based control (Chapter 4) or the measured queue length in rate-based control (Chapter 5). The control signal is ***u***. As suggested in the figure, the delay and the queueing are two major components that are integrated differently along with the network parameters (*N* and *C*, etc.) in the window-based (Chapter 4) or the rate-based controller (Chapter 5), as will be detailed later. Note also that the window-based controllers are using implicit ACK (the ACK has no other information), and rate-based controllers are using explicit ACK (the ACK contains the advertised rate). The symbols and notations introduced above shall pertain for the remainder of the thesis.

2.3 The H_2/H_∞ Controller Model

The design principle of our proposed H_2/H_∞ controller calls for the bounding of different sensitivity functions as a function of *K(s).* The sensitivity function *S(s) = 1/[1+K(s)P(s)]* is the transfer function from the reference input y_{ref} in Fig. 2.3 to the error function (between the plant output ***y*** and the reference input); this is one of the main indicators of the closed-loop performance [Ogat02]. Another indicator is the Complementary Sensitivity function $T(s) = I -$

$S(s) = \frac{K(s)P(s)}{1+K(s)P(s)}$, which is the transfer function between the reference input and plant output. Finally, the Input Sensitivity function $U(s) = K(s)S(s) = \frac{K(s)}{1+K(s)P(s)}$ is the transfer function from the reference input to the plant input ***u***. Let $W_S(s)$, $W_U(s)$ and $W_T(s)$ be the weight functions associated with the sensitivity functions *S(s)*, *U(s)* and *T(s)* respectively. For notation simplicity, we shall sometime express them as W_S, W_U and W_T below without s.

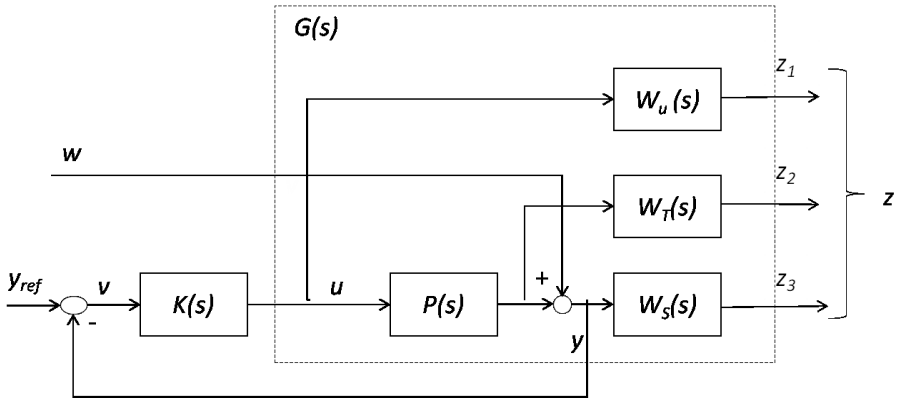


Fig 2.4: The General H_2/H_∞ Control Configuration

Fig. 2.4 has extended the general feedback control of Fig. 2.3 to become the H_2/H_∞ controller. In addition to ***u***, ***y*** and y_{ref} in Fig. 2.3, the input ***w*** is introduced to capture the uncontrolled traffic (such as disturbances and noise) entering the systems. Three weight functions are added: the sensitivity weight $W_S(s)$, the input sensitivity weight $W_U(s)$ and the complementary sensitivity weight $W_T(s)$. They are used to limit the sensitivity of the control system to different parameters monitored by the three sensitivity functions of *S(s), U(s)* and *T(s)* respectively. The components of the output vector ***z*** = [z_1, z_2, z_3] represent the output of these weight functions. As suggested by the dotted box in Fig. 2.4, we have extended the original plant *P(s)* to a generalized plant *G(s)* to facilitate our later analysis and design by including the three sensitivity weight functions.

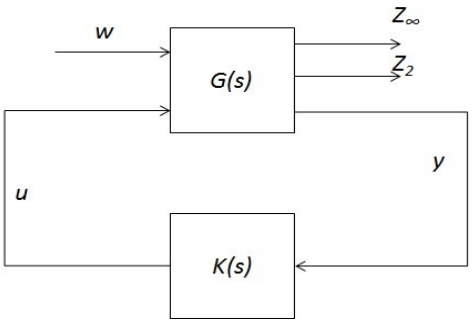


Fig. 2.5: The General Mixed H_2/H_∞ Controller

Fig. 2.5 shows the condensed and rearranged version of Fig. 2.4. Under the H_2/H_∞ formulation of the closed loop function, the weight-control output vector ***z*** in Fig. 2.4 could be re-represented in two channels: the output channel $\boldsymbol{Z_\infty}$ associated with the H_∞ performance and the output channel $\boldsymbol{Z_2}$ associated with the H_2 performance. The inputs to *G(s)* are the uncontrolled traffic ***w*** and the normal input (or control purpose) ***u***.

Next, let $\boldsymbol{E_\infty}$ be the closed-loop transfer functions from ***w*** to $\boldsymbol{Z_\infty}$ in the H_∞-space, and $\boldsymbol{E_2}$ be the closed-loop transfer functions from ***w*** to $\boldsymbol{Z_2}$ which is a transfer function in the H_2-space. If we let $\|\boldsymbol{E}_\infty\|_\infty$ and $\|\boldsymbol{E}_2\|_2$ be their H_∞- norm and H_2-norm, respectively, then according to their definitions [Mack04] in pure frequency response of s=ω, we have

$$\|\boldsymbol{E}_\infty\|_\infty = \max_\omega \bar{\sigma}\left(\boldsymbol{E}_\infty(j\omega)\right) \tag{2.1}$$

and $$\|\boldsymbol{E_2}\|_2^2 = \frac{1}{2\pi}\int_{-\infty}^{+\infty}\left(\sum_{i=1}^{m}\left(\sigma_i\left(\boldsymbol{E_2}(j\omega)\right)\right)^2 d\omega\right) \tag{2.2}$$

where σ(**E**) is the eigen-values of matrix **E**.

Note that there can be different types of the H_2/H_∞ controller depending on the types of limit we want to place on the sensitivity functions. For example, if we do not want to limit the Complementary Sensitivity function, there will be no $W_T(s)$ and the associated output z_2 in Fig. 4. Then, the output vector is simply ***z*** = [z_1, z_3]. By skipping the Complimentary Sensitivity Function in the design procedure, the general plant function *G(s)* now contains only the original plant model *P(s),* and the two weight functions $W_S(s)$ and $W_U(s)$. We shall call this controller the SU controller. Similarly, by not considering the limitation on the input sensitivity function *U(s),* its weight function $W_U(s)$ is not needed. Therefore, the *G(s)* contains two other weight

functions $W_S(s)$ and $W_T(s)$ and the controller will be called the ST controller. In the general form with all three weight functions included, the controller will be called the STU controller.

2.4 Assumptions

Unless otherwise specified, the following assumptions for the remainder of this proposal pertain.

1) Destination router has enough buffers to receive data from source router to approximate no packet loss on this router.
2) The propagation delay and the queueing delay along the date path are two dominant components of the RTT while other components like processing delay of the packets routers or hosts are negligible in comparison.
3) All routers exercise the FIFO (First In First Out) queue discipline.
4) Long-lived flows have infinitely long files to send when the source is active. The flow greedy behavior from this assumption would impose a severe traffic condition to the network so that we could verify the robustness of our new scheme upon the heavy traffic challenges.
5) Network connection breakage rarely happens in order to facilitate the analysis and performance evaluation of our controllers.
6) The size of the advertised window from the receiver is set sufficiently large so that TCP connections are not constrained at the destination.
7) To obtain/design the controller, the values of network parameters, C, N and R_0 can be obtained as discussed earlier.
8) Other general design considerations of QoS and scheduling are not considered in order to focus on the design of the controller. In fact, the prove-in of our controllers would help to enhance these design considerations.

Chapter 3
Designing the H_2/H_∞ Congestion Controller

In this chapter, we give the design principle of three H_2/H_∞ controllers: SU, ST and STU. To make the discussion more concrete, we apply them to the window-based TCP/AQM design. The principles and details of H_2/H_∞ control can be found in many publications such as [Sche95]. For the convenience of the readers, we have provided a brief introduction of the H_x concept and its notation in Appendix B.

3.1 Design Principle

With reference to Section 2.2, the design goal of the H_2/H_∞ controller is to find *K(s)* in order to minimize the criterion γ [Sche05] where

$$\gamma^2 = \alpha\|\boldsymbol{E}_\infty\|_\infty^2 + \beta\|\boldsymbol{E_2}\|_2^2 \qquad (3.1)$$

subject to

$$\|\boldsymbol{E}_\infty\|_\infty \le \delta_0 \quad \text{and} \quad \|\boldsymbol{E_2}\|_2 \le \nu_0$$

where [α , β] are weights to be chosen for $\|\boldsymbol{E}_\infty\|_\infty$ and $\|\boldsymbol{E_2}\|_2$ respectively. For example, having *β=0* gives rise to the H_∞ controller while *α=0* gives rise to the H_2 controller. Having *α=β=1* gives same emphasis to the steady-state performance (due to the H_∞ controller) as well as the transient performance (due to the H_2 controller) The limits $[\delta_0,\ \nu_0]$ are the upper-bounds chosen respectively for $\|\boldsymbol{E}_\infty\|_\infty$ and $\|\boldsymbol{E_2}\|_2$. Larger $[\delta_0,\ \nu_0]$ values mean looser limits on the $\|\boldsymbol{E}_\infty\|_\infty$ and $\|\boldsymbol{E_2}\|_2$ which lead to a less robust system. Readers may find in the Appendix B, a summary of the H_p space, and its reference [Mack04] for further information.

To solve the optimization problem mentioned in eqn. (3.1), we shall use the LMI approach that is often used to solve a convex or quasi-convex optimization problem [BoEl94]. Different efficient numerical methods are also available to solve LMI problem such as those available in MATLAB [Matl13] that we shall use in this thesis. The LMI formulation allows the inequality constraints of an optimization problem to be converted into a numerical problem that can be computed iteratively. Using this formulation [Sche95], we convert and summarize the constraints of the optimization problem in eqn. (3.1) as follows. Appendix J provides more

background on the LMI formulation. Any missing definitions and the physical/performance meaning of parameters can be found in Section 4.2.3.

1) For the H∞ performance, the constraint on $\|\boldsymbol{E}_\infty\|_\infty$ is equivalent to finding a positive definite symmetric matrix $\boldsymbol{X}_\infty$ [Sche95]

2) For the H_2 performance, the constraint on $\|\boldsymbol{E_2}\|_2$is equivalent to finding two symmetric matrices $\boldsymbol{X_2}$ and $\boldsymbol{Q}$ such that $Trace(\boldsymbol{Q}) = \|\boldsymbol{E_2}\|_2^2 < \nu_0^2$.

3) We also need to find a symmetric positive matrix X_{pol} along with two matrices $\boldsymbol{M} = \{\mu_{ij}\}_{1\le i,j\le m}$ and $\boldsymbol{L} = \{\lambda_{ij}\}_{1\le i,j\le m}$to confine the poles to the left half of the complex plane $\mathbb{C}$ as usually expected. Optimization occurs when there is a single matrix X such that $X \coloneqq X_\infty = X_2 = X_{pol}$.

If we let $Y \coloneqq KX$, where K is the to-be-designed controller, and $\delta = \|\boldsymbol{E}_\infty\|_\infty < \delta_0$ then our optimization becomes

$$\text{Minimize } \alpha\delta^2 + \beta Trace(Q)\ over\ X, Q, and\ \delta^2 \tag{3.2}$$

$$\text{satisfying } \begin{bmatrix} A_xX + XA_x^T + B_2Y + Y^TB_2^T & B_1 & XC_1^T + Y^TD_{\infty 2}^T \\ B_1^T & -I & D_{\infty 1}^T \\ C_\infty X + D_{12}Y & D_{\infty 1} & -\delta^2 I \end{bmatrix} < 0$$

$$\begin{bmatrix} Q & C_2X + D_{22}Y \\ XC_2^T + Y^TD_{22}^T & X \end{bmatrix} > 0$$

$$\left[\lambda_{ij} + \mu_{ij}(A_xX + B_2Y)X + \mu_{ij}(XA_x^T + Y^TB_2^T)\right]_{1\le i,j\le m} < 0$$

The corresponding controller of the optimal solution of our problem $(Y^*, X^*, Q^*, \delta^*)$ is then given by

$$K^* = Y^*(X^*)^{-1} \tag{3.3}$$

and this controller can guarantee that

$$\|\boldsymbol{E}_\infty\|_\infty \le \delta^* \quad \text{and} \quad \|\boldsymbol{E_2}\|_2 \le \sqrt{Trace(Q^*)} \tag{3.4}$$

Eqn. (3.4) suggests that the optimization problem can be decomposed into 2 optimization sub-problems/objectives, minimizing $\delta = \|\boldsymbol{E}_\infty\|_\infty$ and minimizing $Trace(Q) = \|\boldsymbol{E_2}\|_2$ simultaneously. Therefore, the LMI problem introduced in eqn. (3.2) can be solved as a linear objective minimization problem under LMI constraints [NoGa94]. The Projective Algorithm

method, which is a numerical and iterative solution for LMI problems, has been implemented in *Robust Control Toolbox*, MATLAB [Matl13] and is used to solve eqn. (3.2) to find the controller.

The design principle calls for the bounding of different sensitivity functions as a function of *K(s)*. As introduced in Section 2.3, $W_S(s)$, $W_U(s)$ and $W_T(s)$ are the weights associated with the Sensitivity function *S(s)*, Input Sensitivity function *U(s)* and the Complementary Sensitivity function, *T(s)* respectively. Loosely speaking, one uses $1/|W_U|$ to restrict input signal while using the interaction of $1/|W_S|$ and $1/|W_T|$ to control the steady-state error and transient response as the minimum bandwidth frequency ω_B and its roll-off slope are affected in the bound specification [SkPo05]. Our next job is to choose proper weight functions with respect to minimizing γ (and therefore to have robust and good performance in the transient responses and small steady state error).

3.2 Weight Selection of the Three Sensitivity Functions

In general, a more robust design calls for smaller sensitivity functions. According to the definition of the H_∞ norm for a robust controller, one would bound the magnitude of the sensitivity weight function $W_S S(s)$ to be less than one. Physically, it means that the shape of magnitude curve of the sensitivity function *S(s)* is upper bounded by $1/|W_S|$. Similarly, we could place an upper bound $1/|W_U|$ on the magnitude of the input sensitivity function *U(s)* to restrict the magnitude of the plant input signals to achieve robustness. Likewise, the magnitude of the $W_T T(s)$ would be bound to be less than one. It can also be shown [SkPo05] that in the real scenario when γ_{min} (the minimum of γ) has any value other than 1, the magnitude of *S* must be bounded by $\gamma_{min}/|W_S|$. Likewise, *U(s)* upper bounded by $\gamma_{min}/|W_U|$ and *T(s)* upper bounded by $\gamma_{min}/|W_T|$. The following are further requirements on choosing the weight functions properly in order minimizing γ (and therefore to have robust and good performance in the transient responses and small steady state error).

3.2.1 Sensitivity Weight Function

The sensitivity function $S(s) = \frac{1}{K(s)P(s)}$ is the transfer function from the reference input to the

error signal (which is the difference between the plant output and the reference input). It is a very good indicator of closed-loop performance under various disturbance parameters. The smaller the function magnitude, the smaller the sensitivity (and hence more robust) to those disturbance parameters. As one approach, we can consider the frequency response when s= ω. According to the requirement, we have

$$|W_S S| < 1 \Leftrightarrow, |S(j\omega)| < \frac{1}{|W_S(j\omega)|}, \forall\omega \quad (3.5)$$

We also would like to use an asymptotic upper bound [SkPo05] given by

$$W_S = \frac{s/M + \omega_B}{s + \omega_B A} \quad (3.6)$$

where *A* is the maximum steady-state tracking error, *M* is the upper bound for the maximum peak magnitude of *S(jω)*, and ω_B is the cross-over frequency of unit magnitude. One can see the upper bound on $|S(j\omega)|$ is equal to *A* at low frequency, and equal to *M* at high frequency. The frequency ω_B can be used to approximate the bandwidth requirement [SkPo05]. This leads us to the following consideration in selecting the sensitivity weights. Note that selecting *A*<<1 ensures the approximation of the integral action with $S(0) \approx 0$.

3.2.2 Input Sensitivity Weight Function

Input sensitivity function is a characteristic function of any system. One can verify that the input sensitivity function $U(s) = K(s)S(s) = \frac{K(s)}{1+ K(s)P(s)}$ is the transfer function from the reference input ***w*** to the plant input ***u*** in Fig. 2.4. We can limit the plant input by choosing W_U with a value less than the magnitude of 1/*U(s)*, i.e., *U(s) is* upper bounded by $1/|W_U|$. A reasonable choice for the input weight is

$$W_u = \rho \quad (3.7)$$

where ρ is a constant number.

3.2.3 Complementary Sensitivity Weight Function

As introduced before, the complementary sensitivity is $T(s) = I - S(s) = \frac{K(s)P(s)}{1+K(s)P(s)}$. The specification $|W_S S| < 1$ puts a lower bound on the bandwidth, but not an upper one, and nor

does it allows us to specify the roll-off of L (i.e., the open-loop transfer function) above the bandwidth. To obtain an upper bound and a proper roll-off, we can specify an upper bound $1/|W_T|$ on the magnitude of $T(s)$ to make sure that L rolls off sufficiently fast at high frequencies. Therefore, we want $T(s)$ to be small at high frequencies to reduce sensitivity to noise and uncertainty. In general, we can specify the $W_T(s)$ transfer function by

$$W_T(s) = \frac{a_1 + a_2 s}{1 + a_3 s} \tag{3.8}$$

where a_1 is a small constant to ensure that $|W_T|$ is small at low frequencies. We would also choose the other two constants such that $a_3 << a_2$ in order to guarantee $|W_T|$ to be large at high frequencies.

3.3 Design Procedure for the H_2/H_∞ Controllers

Based on the above formulation, Fig. 3.1 below shows the general procedure in designing an H_2/H_∞ STU controller using the *Robust Control*-toolbox in MATLAB.

Step#	Step description	YES	NO
1	Define the plant model $P(s)$ Assign $[\alpha, \beta]$ and initialize $[\delta_0, \nu_0]$ to some relatively large numbers	Next step	
2	Initialize W_S, W_T and W_U based on eqn. (3.6), (3.7), (3.8)	Next step	
3	Find the $K(s)$ by Solving Iterative Optimization Problem in eqn. (3.1) using the iterative Projection Algorithm from MATLAB	Next step	
4	Feasible result	Go to 6	Go to 5
5	Increase $[\delta_0, \nu_0]$	Go to 3	
6	Check the Robustness Condition, $\|(S(j\omega)\|<\gamma_{min}/\|W_S\|$ and $\|U(j\omega)\|<\gamma_{min}/\|W_U\|$, $\|T(j\omega)\|<\gamma_{min}/\|W_T\|)$ and the positivity of the Phase Margin and the Gain Margin[1]	Next step	Go to 8
7	Check the reasonability of the performance of the controller in a Simulator	Go to 9	Go to 8
8	Change W_S, W_T and W_U parameters[2]	Go to 3	
9	Stop and $K(s)$ is the Optimal Controller Associated with the final W_S, W_T and W_U.		

[1] Note that positive Gain Margin and Phase Margin shows the controller is stable.
[2] W_S, W_T and W_U are changed one at a time. For example, W_S is changed while W_U and W_T are fixing. In addition, within the same procedure for each weight function, one parameter is changed while others (if any) are fixed.

Fig 3.1: Design Procedure of the H_2/H_∞ Controller

In the design procedure, a controller design is called feasible if the three optimization conditions mentioned Section 3.1 or the equivalent eqn. (3.2) are satisfied and when a matrix X (and therefore the controller K) is found. The optimization problem is solved iteratively; the

code terminates when the object term in eqn. (3.2) which is $\alpha\delta^2 + \beta Trace(Q)$ has not decreased by more than a desired relative accuracy (say 10^{-2}) during the last 10 iterations. In addition, the performance of the controller is expected to show a convergence trend in the transients (e.g., the rise/fall times) and the steady state response (e.g., the queue peak value) of the router 1 buffer. Since we have not found any good reasons to bias against anyone of the two norms, we choose the weighting [α, β]=[1, 1] for the H_∞-norm and H_2-norm.

As mentioned earlier in Chapter 2, there can be different types of the H_2/H_∞ controller i.e. the STU, SU, ST controllers, depending on the types of limit we want to place on the sensitivity functions. Therefore, certain weight function may be omitted (e.g. no W_T is need in Fig. 3.1 if SU controller is called for).

3.4 Example

As an example of H_2/H_∞ controller, we discuss briefly the STU controller here (see more details in Chapter 4 later). We use *Robust Control*-toolbox in MATLAB by assigning the same weights of [α, β] = [1, 1] to $\|\boldsymbol{E}_\infty\|_\infty$ and$\|\boldsymbol{E_2}\|_2$, and choosing some relatively large numbers for δ_0 , ν_0 (e.g., $[\delta_0, \nu_0]$ = [200, 200]) in order for the iteration process converge to the optimum result. After some iterations (about 10), the weight functions are determined to be $W_U(s) = 10^5, W_S(s) = \frac{s+11}{2s+11\times10^{-7}}$ and $W_T(s) = \frac{0.001+6s}{1+0.01s}$.

In addition, the final STU controller is obtained as

$$K(s) = \frac{1.739\times10^{-10}\mathrm{s}^4 + 3.563\times10^{-8}\,\mathrm{s}^3 + 4.314\times10^{-7}\,\mathrm{s}^2 + 1.243\times10^{-6}\,\mathrm{s} + 3.699\times10^{-7}}{9.648\times10^{-5}\,\mathrm{s}^5 + 0.0198\,\mathrm{s}^4 + 0.253\,\mathrm{s}^3 + 0.8189\,\mathrm{s}^2 + 0.5147\,\mathrm{s} + 2.648\times10^{-7}}$$

Using the above controller function, we can determine H_∞-norm of the closed-loop transfer functions from $\boldsymbol{w}$ to $\boldsymbol{Z_\infty}$ is $\|E_\infty\|_\infty = 21.7344$ and H_2-norm of closed-loop transfer functions from $\boldsymbol{w}$ to $\boldsymbol{Z_2}$ is $\|E_2\|_2 = 20.8166$. These norms give the minimum of the mixed H_2/H_∞ norm at $\gamma_{min} = 30.0951$.

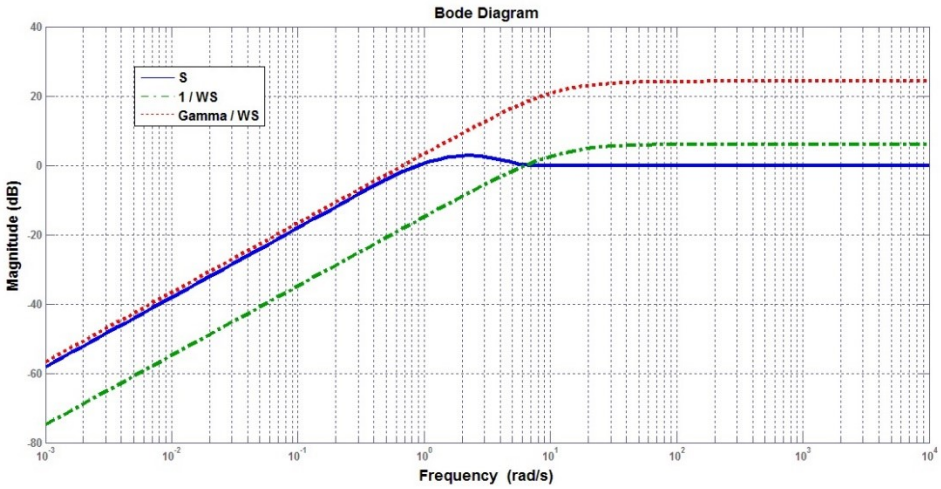


Fig. 3.2: Bode Diagram of *S*, $1/W_S$ and γ_{min}/W_S for the STU Controller

Fig 3.2 presents the Bode diagrams of the sensitivity function *S(s)* (indicated by the solid blue curve), the inverse of the sensitivity weight function $1/W_S$ (the dotted red curve) and γ_{min}/W_S (the dashed green curve) of the STU controller in order to verify the feasibility of our weight functions selected for $W_s(s)$ (see Section 4.3.1.1 later). Note that *S(s)* is small at low frequencies, which means the sensitivity of the system output to disturbance is small at low frequencies, and likewise the error between system output and the reference input is small. Conversely, in order to attenuate the measurement noise and reduce the influence on the tracking error, we desire *S(s)* to be around 1 at high frequencies which means the magnitude of the open loop transfer function is large at high frequencies. The figure shows that the magnitude of *S(s)* reaches the asymptotic value of 1 (0dB) at high frequencies which is the desirable characteristic of sensitivity function [DoBi10]. In addition, Fig 3.2 has verified that $|S(j\omega)|$ is upper bounded by $\gamma_{min}/|W_S|$ which meets one of the requirements in the STU controller design.

Similarly, Fig. 3.3 below represents the Bode diagrams of the input sensitivity function *U(s)*, the inverse of the sensitivity weight function $1/W_U$ and γ_{min}/W_U of the STU controller It is shown that $|U(j\omega)|$ is upper bounded by $\gamma_{min}/|W_U|$, and hence the correctness of the parameter selection is verified for W_U.

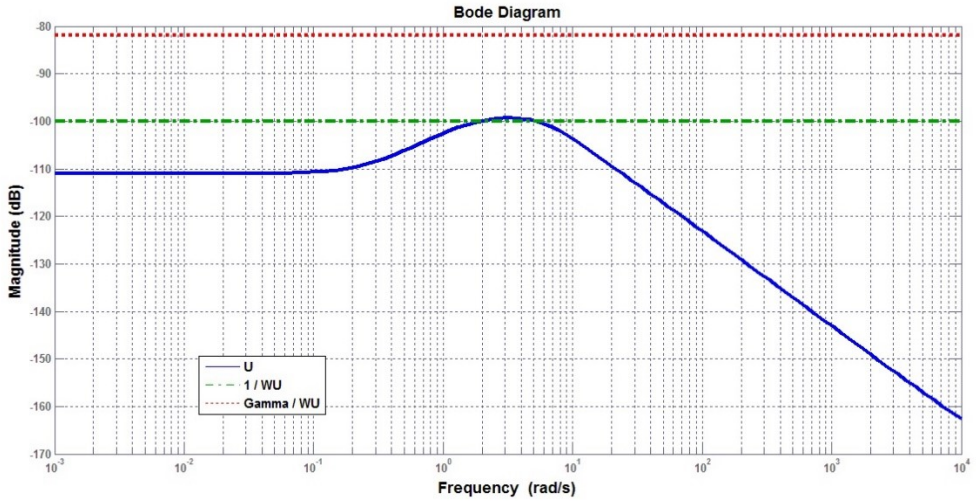


Fig. 3.3: Bode Diagram of U, $1/W_U$ and γ_{min}/W_U for the STU Controller

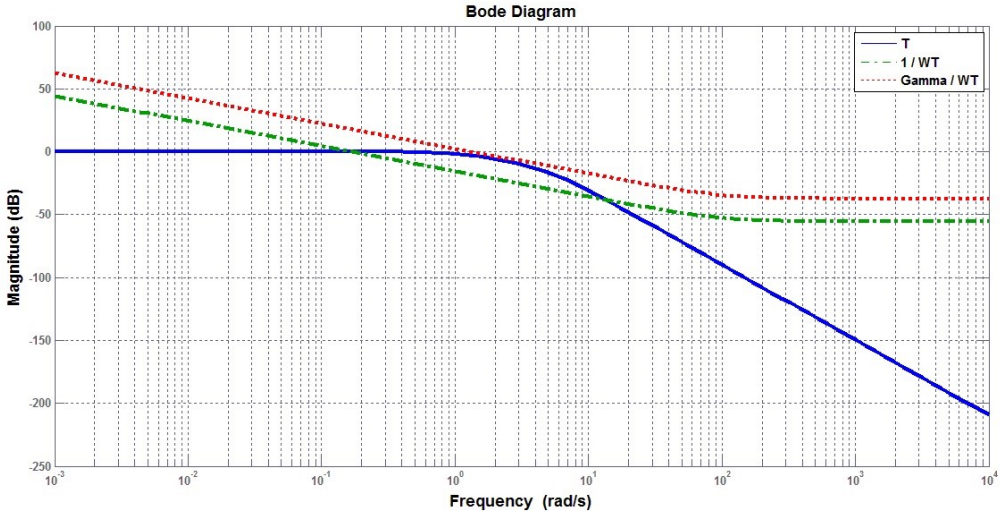


Fig. 3.4: Bode Diagram of T, $1/W_T$ and γ_{min}/W_T for the STU Controller

Fig 3.4 presents the Bode diagrams of the complementary sensitivity function T, the inverse of the complementary sensitivity weight function $1/W_T$ and γ_{min}/W_T. It is shown that T is 0 dB at low frequencies, and drops off at high frequencies, which reduces the sensitivity to noise and uncertainty at high frequencies. In addition, we have mentioned that, $|T|$ is upper bounded by $1/|W_T|$. Fig. 3.4 also verifies that $|T|$ is upper bounded by $\gamma_{min}/|W_T|$. Similar Bode diagrams for the SU and the ST controllers can be found in Appendix C.

3.5 Concluding Remarks

In this chapter, we have given the principle of designing the H_2/H_∞ controller. The feature of this controller is the addition of H_2 formulation, which has improved the transient performances in addition to the robustness feature of H_∞ controller. Our design uses the different sensitivity functions as well as their weight functions to monitor and control the closed-loop dynamics and to bind them to within reasonable performances. A convex optimization problem as an LMI problem has been solved in order to find the controller. An iterative numerical method has been used to solve our LMI problem, which can be considered as a linear objective minimization.

The design procedure and the discussion here will apply in future chapters but the same details may be omitted there for conciseness and clarity purposes.

Chapter 4
Window-Based H_2/H_∞ Controllers

In this chapter, we shall use the H_2/H_∞ approach to design our SU, ST, and STU controllers. These are the window-based controllers designed for the TCP/AQM networks. Since there can be different combinations of choosing weight functions, Section 4.2 gives the H_2/H_∞ controller design using the SU formulation. Similarly, we have Section 4.3 for the ST controller and Section 4.4 for the STU controller. Finally, their performances will be evaluated and compared with some existing controllers in Section 4.5 for known values of network parameters. Their stability is also analyzed in the individual sections.

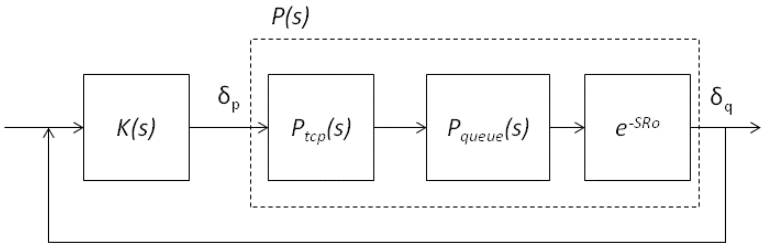


Fig. 4.1: TCP/AQM Feedback Control

4.1 Window-Based Control System Model for the TCP/AQM Networks

Fig. 4.1 shows more specific details of Fig. 2.3 for our window-based control model. The plant transfer function *P(s)* consists of three parts: the block $\boldsymbol{P_{tcp}(s)}$ is the plant transfer function capturing the TCP dynamic in the TCP/AQM network; the block $\boldsymbol{P_{queue}(s)}$ is the plant transfer function capturing the queueing behavior; and the block $\mathbf{e}^{-\mathbf{sR_0}}$ is the delay of the system where $\boldsymbol{R_0}$ represents the round trip time, RTT. The controller *K(s)* is designed to control the variation of the queue length δ_q in the router (in addition to maintaining the router queue length at a desired level) by controlling the packet drop probability. As shown in the figure, the controller determines and passes the drop probability variation δ_p to the plant *P(s)*, and the queue size variation δ_q arising from the plant operation is fed back to the controller.

As introduced in Section 2.2, *C* is the link capacity (packets/sec); *N* is load factor (number of TCP sessions); and $\boldsymbol{R_0}$ is the average delay of all flows. Then according to the TCP window size

control dynamics and AQM queue dynamics, we can have the following approximation.

$$P_{tcp}(s) = \frac{\frac{R_0 C^2}{2N^2}}{s+\frac{2N}{R_0^2 C}}; \qquad P_{queue}(s) = \frac{\frac{N}{R_0}}{s+\frac{1}{R_0}}. \tag{4.1}$$

Readers may refer to Appendix A for more details. Note that the packet drop probability has an equilibrium point at $p_0 = 2N^2/(R_0 \times C)^2$ [HoMi02]. Obviously,

$$P(s) = P_{tcp}(s)P_{queue}(s)e^{-sR_0} \tag{4.2}$$

If we further linearize e^{-sR_0} by the first order Pade approximation [TuBa07] where $e^{-sR_0} \approx \frac{1-(R_0/2)s}{1+(R_0/2)s}$, we can obtain

$$P(s) = \frac{\frac{-C^2}{2N}s+\frac{C^2}{R_0 N}}{s^3+\left(\frac{2N}{R_0^2 C}+\frac{3}{R_0}\right)s^2+\left(\frac{6N}{R_0^3 C}+\frac{2}{R_0^2}\right)s+\frac{4N}{R_0^4 C}}. \tag{4.3}$$

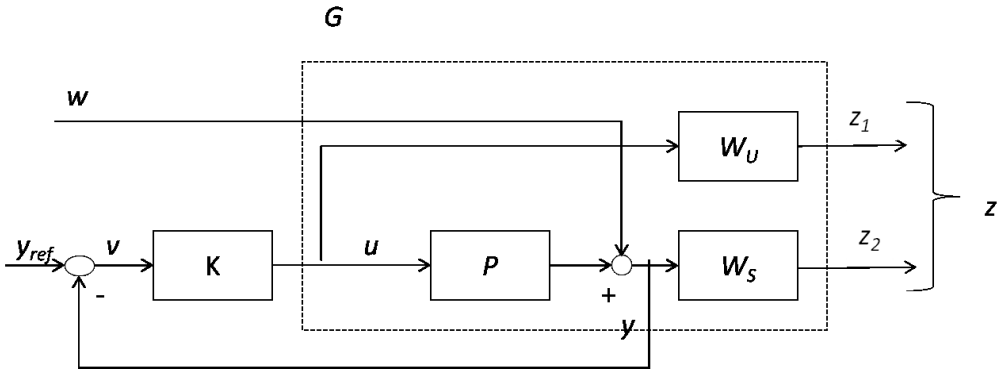


Fig. 4.2: General Configuration for the SU Controller

4.2 The SU Controller

As we mentioned earlier, there is no limit on the Complementary Sensitivity function in the SU controller. Therefore, there will be no $W_T(s)$ and the associated output z_2 in Fig. 2.4. Then, the output vector is simply $\boldsymbol{z} = [z_1, z_3]$. By skipping the Complimentary Sensitivity Function in the design procedure, the general plant function $G(s)$ now contains only the original plant model $P(s)$, and the two weight functions $W_s(s)$ and $W_u(s)$. Fig. 4.2 shows the general configuration of the SU controller. With reference to the given control system model in Fig. 4.1, the weighted drop probability variation δ_P is the exogenous output z_1, and the weighted queue size error δ_q (queue size variation) is the exogenous output z_3 in Fig. 4.2. Indeed, the queue size error is also the controller input $\boldsymbol{v}$ in Fig. 4.2 and the drop probability variation is the control signal $\boldsymbol{u}$. In addition, $\boldsymbol{w}$ represents the disturbance and noise and y_{ref} indicates reference (desire) queue

size in the TCP/AQM system.

4.2.1 Controller Design

To find the generalized plant *G(s)* in Fig 4.2, we need to specify the weight functions $W_S(s)$ and $W_U(s)$ based on the information given in Section 3.2. Unfortunately, we cannot obtain closed-form equations for those weights and parameters of lower- and upper-bounding. Therefore, we have to resort to simulation to obtain their values. Section 4.2.2 provides a design example on the selection of these weight functions and its impacts on the performance of the TCP/AQM system, and hence illustrating how we can arrive at our choice.

The design procedure for SU controllers would be similar to Fig. 3.1 except for Step-2, by not initializing W_T. Also, the condition $|T(j\omega)|<\gamma_{min}/|W_T|$ in Step-6 need not to be checked anymore. In general, we can skip any steps where W_T is involved as there is no W_T in the SU controller.

4.2.1.1 Design Example

As an example for designing as SU control for the TCP/AQM network, the parameter values we use are *N*= 100 greedy ftp sources using TCP sessions along with 20 http sources, a round trip time R_0=0.25s, and the bandwidth of the bottleneck link at *C*=9708 packets/second (which gives us a T3 rate of 44,736,000bps using an average packet size of 576 bytes). Hence, we can obtain [HoMi02]

$$P(s) = \frac{471226.32}{(s+0.3296)(s+4)} e^{-0.25s} \qquad (4.4)$$

We now follow the iterative design procedure of Fig. 3.1 with the changes/omissions mentioned in Section 4.2.1. We use OPNET simulations to carry out Step 7. We also use *Robust Control*-toolbox in MATLAB by assigning the same weights of [α, β] = [1, 1] to $\|\boldsymbol{E}_\infty\|_\infty$ and $\|\boldsymbol{E_2}\|_2$, respectively, and by choosing $[\delta_0, \nu_0]$ = [200, 200] which are large enough for the iteration process converge to the optimum result. When the results converge in about 10 iterations, and the weight functions $W_U(s)$ and $W_S(s)$ are determined to be W_U =10^5 and $W_s(s) = \frac{s+11}{2s+11\times10^{-7}}$. The final SU controller is obtained as

$$K(s) = \frac{(0.2747\text{s}^3+3.401\text{s}^2+10.05\text{s}+3.36)\times10^{-6}}{0.003583\ \text{s}^4+0.06378\ \text{s}^3+\ 0.3242\ \text{s}^2+0.9438\ \text{s}-4.991\times10^{-7}} \quad (4.5)$$

Using the above controller function, we can determine the H_∞-norm of the closed-loop transfer functions from $\boldsymbol{w}$ to $\boldsymbol{Z_\infty}$ to be $\|\boldsymbol{E}_\infty\|_\infty = 5.4244$. Similarly, the H_2-norm of the closed-loop transfer functions from $\boldsymbol{w}$ to $\boldsymbol{Z_2}$ is $\|\boldsymbol{E_2}\|_2 = 3.3657$. These norms give the minimum of the mixed H_2/H_∞ norm, i.e. $\gamma_{min} = 6.3838$. By using the Bode diagrams, one can show that our designed W_S and W_U are feasible functions as they meet the expected criteria in Section 3.2. Readers may refer to Appendix C for more information.

4.2.2 Effect of the Design Parameters

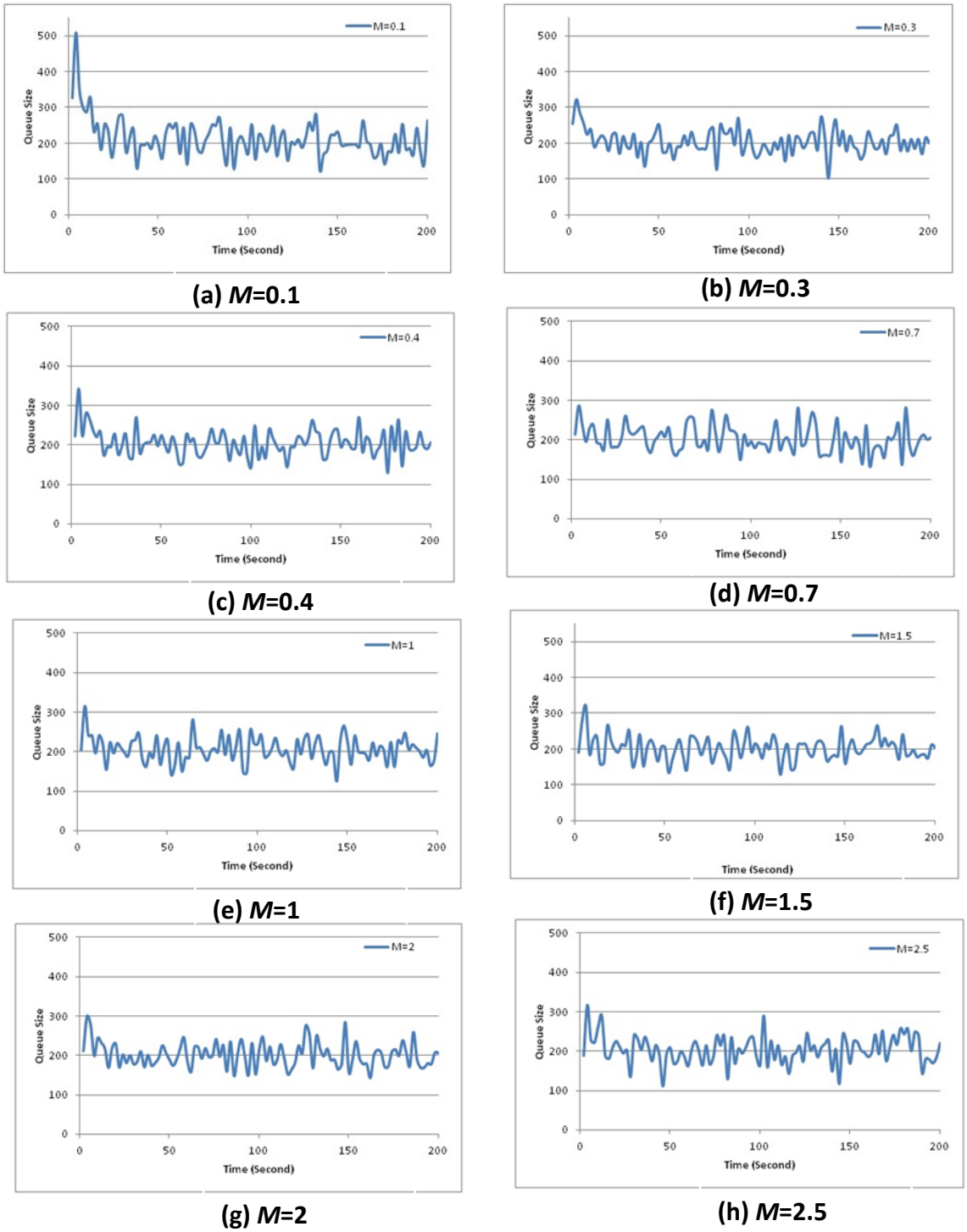
We shall now use the same example as in Section 4.2.1.1 to investigate the impacts of the weight functions of W_U and W_S on the system performance. Specifically, we vary the parameters ω_B, M or A *in* W_S while fixing W_U. Then we vary W_U while fixing W_S. We also compare different designed controllers under different performance measures whose definitions can be found in the Performance Evaluation section later. Keep in mind that we have a new controller every time the value of a parameter changes.

4.2.2.1 Effect of Varying W_S

We shall fix W_u at 10^{-5} and vary the parameters in $W_S = \frac{s/M+\omega_B}{s+\omega_B A}$ one at a time.

Table 4.1: Different SU Controllers Specification by Varying *M*

W_U=10^5, ω_B =6.5, A=10^{-5}	γ_{min}	$\|E_\infty\|_\infty$	$\|E_2\|_2$	$e(\infty)$	PM, GM
M=0.1	13.4260	12.8601	3.8569	1.1388×10^{-4}	77.1°, 15.4dB
M=0.3	8.2393	7.3393	3.7445	5.6825×10^{-5}	63.7°, 9.8dB
M=0.4	7.7880	6.8314	3.7395	5.4389×10^{-5}	61.2°, 9.2dB
M=0.7	7.3550	6.3314	3.7429	4.5666×10^{-5}	58.6°, 8.3dB
M=1.5	7.1770	6.1142	3.7584	4.5199×10^{-5}	57.1°, 7.9dB
M=2	7.1525	6.0857	3.7581	3.7710×10^{-5}	57.3°, 7.6dB



(a) *M*=0.1 (b) *M*=0.3

(c) *M*=0.4 (d) *M*=0.7

(e) *M*=1 (f) *M*=1.5

(g) *M*=2 (h) *M*=2.5

Fig. 4.3: Instantaneous Queue Size of the SU Controllers with W_U=10^5, ω_B= 6.5 and *A*= 10^{-5} but Different *M* Values

a) Varying *M*

Table 4.1 gives γ_{min} as a function of *M* while fixing W_U, ω_B and *A* at 10^5, 6.5 and 10^{-5} respectively. It shows different controllers specifications including γ_{min} (the minimum mixed

H_2 /H_∞-norm), $\|\boldsymbol{E}_\infty\|_\infty$ (the H_∞-norm of closed-loop transfer functions from ***w*** to ***z***), $\|\boldsymbol{E_2}\|_2$ (the H_2-norm of closed-loop transfer functions from ***w*** to ***z***), the steady state error $e(\infty)$, the phase margin PM, and the gain margin GM. By increasing *M*, $\|\boldsymbol{E}_\infty\|_\infty$, $\|\boldsymbol{E_2}\|_2$ and γ_{min} all decrease and give rise to a more robust controller. While steady state error also decreases w.r.t. *M*, there is no significant effect on $e(\infty)$ for *M* greater than 0.1. Moreover, all the controllers are stable as shown in terms of their positive phase margin and gain margin.

Fig. 4.3 illustrates the instantaneous queue size for the SU controller simulated in OPNET for *M* ranging from *M*=0.1 to 2. It is shown that *M*=0.1 gives the longest rise/fall time[3] T_r which is equal to 18 second. In addition, the queue size rises to 509 packets during the initialization of the system before settling back to the target queue size 200 packets. The mean queue length in this case is 214 packets and the standard deviation is σ_q=53 packets.

Table 4.2: Performance of SU Controllers with Different *M*

W_U=10^5, A=10^{-5}, ω_B =1.5	T_r (sec)	Peak Value (Packet)	μ_q (Packet)	σ_q(Packet)
M=0.1	18	509	219	53
M=0.3	14	322	203	34
M=0.4	16	343	204	33
M =0.7	6	286	203	33
M=1	9	315	205	32
M=1.5	7	319	201	32
M=2	7	300	202	31
M=2.5	13	317	203	35

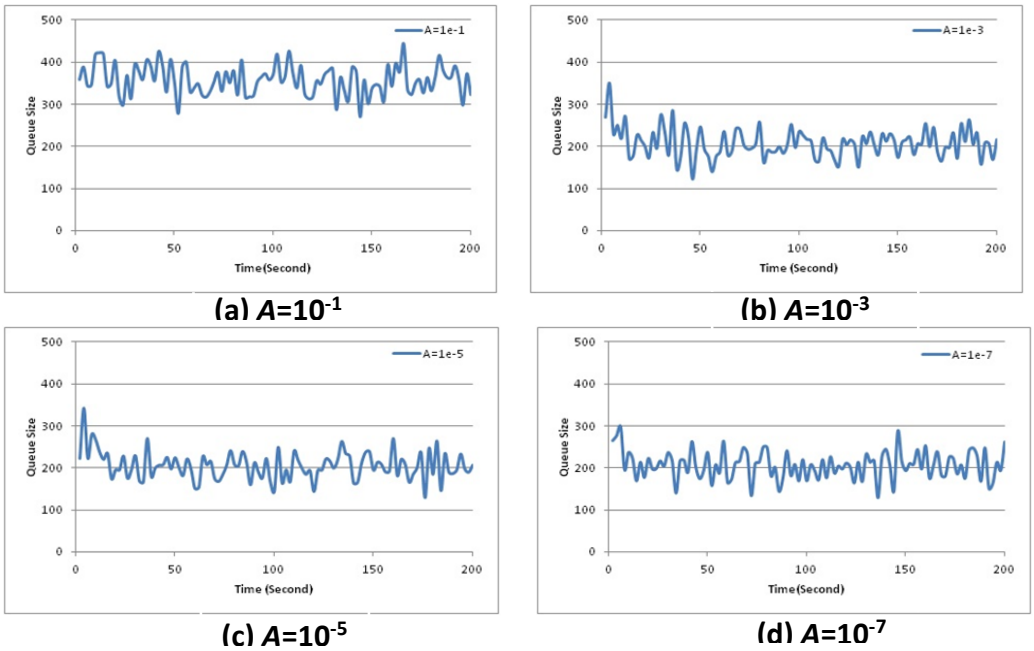
Table 4.2 gives different performance measures including T_r (rise/fall time), peak value, μ_q (mean queue length), σ_q (queue length standard deviation) as a function of *M*. It is shown that *M*=0.1 gives the longest rise/fall time T_r (18 seconds). Note that the target queue size is 200 packets. In addition, the queue size rises to 509 packets during the initialization of the system before settling back to the target queue size of 200 packets. The mean queue length in this case is 214 packets and the standard deviation is σ_q=53 packets. One can also see that the T_r, peak

[3] Rise time is the time required for the signal to go from 10% to 90% of the final value [DoBi10]. The fall time is the time that signal travels from $F + 0.9 \times (P - F)$ to $F + 0.1 \times (P - F)$ where F is the final steady state value and P is the peak value.

value, μ_q, σ_q are not monotonically decreasing w.r.t. *M*. The best results with the smallest T_r and peak value are achieved at *M*=0.7 and 2. Of these two M values, the *M*=2 case has a smaller mean queue size. Its standard deviation is also smaller due to smaller fluctuations. Therefore, we choose *M*=2 for our final *SU* controller.

Table 4.3: Different SU Controller Specifications by Varying *A*

W_U =10^5, ω_B =6.5, *M*=0.4	γ_{min}	$\|E_\infty\|_\infty$	$\|E_2\|_2$	$e(\infty)$	PM, GM
A= 10^{-1}	5.6543	5.2235	2.1647	0.4127	125°, 13.3dB
A=10^{-3}	7.7621	6.8138	3.7177	5.6449×10^{-3}	61.6°, 9.38dB
A=10^{-4}	7.7855	6.8294	3.7381	5.6324×10^{-4}	61.3°, 9.31dB
A=10^{-5}	7.7880	6.8314	3.7395	5.4389×10^{-5}	61.2°, 9.32dB
A=10^{-7}	7.7878	6.8329	3.7364	4.2336×10^{-7}	61.0°, 9.26dB



(a) *A*=10^{-1} **(b) *A*=10^{-3}**

(c) *A*=10^{-5} **(d) *A*=10^{-7}**

Fig. 4.4: Instantaneous Queue Size of the SU Controllers with W_U=10^5, ω_B= 6.5 and *M*= 0.4 but Different *A* Values

b) Varying *A*

Table 4.3 gives different controllers specifications including γ_{min}, $\|\boldsymbol{E}_\infty\|_\infty$, $\|\boldsymbol{E}_2\|_2$, $e(\infty)$, the phase margin PM, and the gain margin GM as a function of *A* while fixing W_U, ω_B and *M* at 10^5, 6.5 and 0.4, respectively. By increasing *A* from 10^{-7}, $\|\boldsymbol{E}_\infty\|_\infty$, $\|\boldsymbol{E}_2\|_2$ and γ_{min} all decrease but

not significantly. They are more or less constant except for A=10^{-1} which has significantly smaller values for γ_{min} , $\|\boldsymbol{E}_{\infty}\|_{\infty}$ and $\|\boldsymbol{E_2}\|_2$. However, it has significant larger $e(\infty)$. This reconfirms the discussion for eqn. (3.6) in Section 3.2.1 that A should be chosen to be much smaller than 1 in order to have a small steady state error. As seen, the smallest $e(\infty)$ occurs at A=10^{-7}. Moreover, all the controllers as shown are stable in terms of their positive phase margin and gain margin.

Table 4.4: Performance of SU Controllers with Different *A*

W_U=10^5, M=0.4, ω_B =6.5	T_r (Second)	Peak Value (Packet)	μ_q (Packet)	σ_q(Packet)
A=10^{-1}	NA	445	357	35
A=10^{-3}	12	350	203	34
A=10^{-5}	16	343	204	33
A=10^{-7}	7	297	206	33

Table 4.4 shows the performance measures of T_r, peak value, μ_q and σ_q as a function of A. It shows that for A=10^{-1}, system cannot reach the target value of 200 packets. As A decreases, the peak value decreases while μ_q increases, both insignificantly. The best results with the smallest T_r (7 seconds) and peak value (297 packets) are achieved at A=10^{-7}. Since the μ_q and σ_q values at A=10^{-7} are more or less the same as those at A=10^{-3} and 10^{-5}, we would like to choose A=10^{-7} for our final SU controllers.

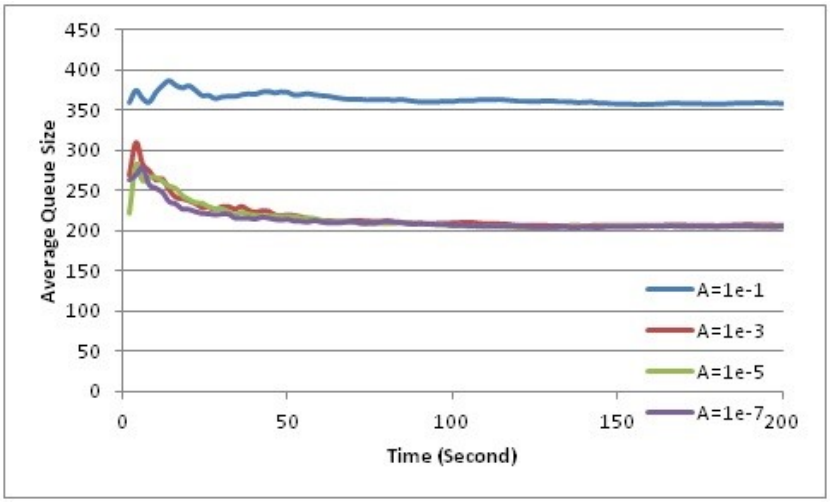


Fig. 4.5: Average Queue Size of the SU Controllers with W_u=10 ^{5}, ω_B= 6.5 and M= 0.4 but Different A Values

Fig. 4.5 is the time evolution of the average queue size for different values of *A* ranging from $A=10^{-1}$ to 10^{-7}. It shows that for $A=10^{-1}$, the average queue size levels off at 357 packets with a steady state error of about 157 packets. The figure also shows that the average queue size of other A values has almost the same steady state packet size which are different from that of $A=10^{-1}$.

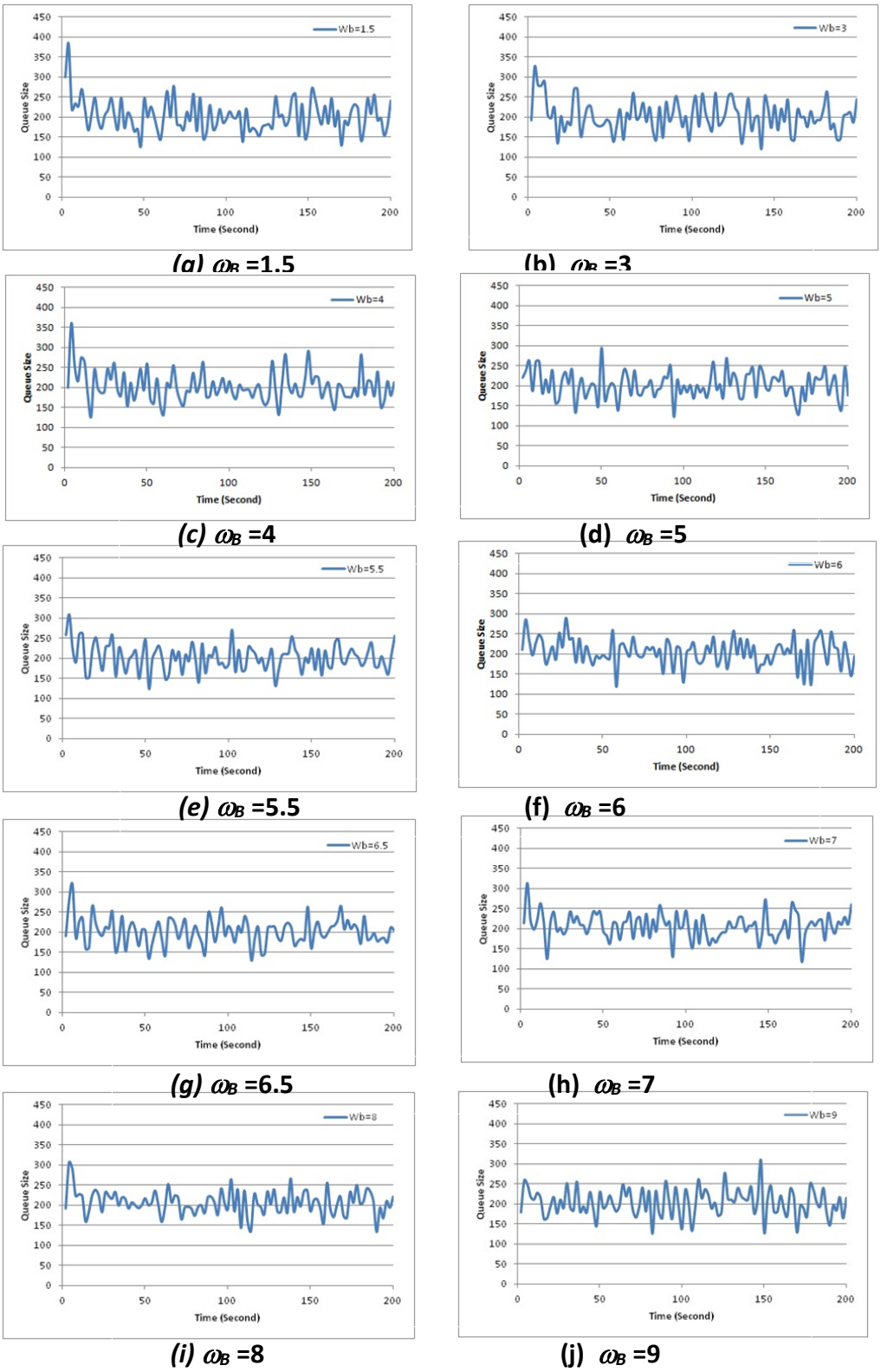
Table 4.5: Different SU Controllers Specification by Varying ω_B

$W_U=10^5$, $A=10^{-5}$, $M=1.5$	γ_{min}	$\|E_\infty\|_\infty$	$\|E_2\|_2$	$e(\infty)$	PM, GM
ω_B =1.5	2.8355	2.3893	1.5269	1.3226×10^{-5}	57.7°, 9.7dB
ω_B =3	4.2983	3.6273	2.3061	8.2669×10^{-6}	57.0°, 8.9dB
ω_B =5.5	6.4016	5.4447	3.3668	3.3706×10^{-5}	57.3°, 8.19dB
ω_B =6.5	7.1770	6.1142	3.7584	4.5199×10^{-5}	57.1°, 7.9dB
ω_B =7	7.5542	6.4408	3.9474	4.9874×10^{-5}	57.0°, 8.1dB
ω_B =8	8.2889	7.0803	4.3097	4.9912×10^{-5}	57.5°, 7.7dB
ω_B =9	8.9998	7.6974	4.6633	5.9112×10^{-5}	57.4°, 7.6dB

c) Varying ω_B

Table 4.5 gives different controller performances of γ_{min}, $\|\boldsymbol{E}_\infty\|_\infty$, $\|\boldsymbol{E}_2\|_2$, $e(\infty)$, the phase margin PM, and the gain margin GM as a function of ω_B while W_U, *A* and *M* are fixed at 10^5, 10^{-5} and 1.5, respectively. By increasing ω_B, one can see that $\|\boldsymbol{E}_\infty\|_\infty$, $\|\boldsymbol{E_2}\|_2$ and γ_{min} all increase and give rise to a less robust controller. Additionally, the steady state error $e(\infty)$ increases w.r.t ω_B when ω_B is greater than 3; At ω_B=3, $e(\infty)$ has the smallest value. However, $e(\infty)$ does not increase significantly. Moreover, all the controllers are stable as shown in terms of their positive phase margin and gain margin.

Fig. 4.6 shows the instantaneous queue size for the H_2/H_∞ controller simulated in OPNET for ω_B ranging from ω_B =1.5 to 9. It is shown that ω_B =1.5 and 9 give the longest rise/fall time T_r (defined in Section 4.2.2.1.a) which is equal to 14 second. In addition, the queue size rises to 384 and 309 packets, respectively during the initialization of the system before settling back to the target queue size 200 packets. The mean queue length in this case is 203 packets for both values and the standard deviation is σ_q=40 and 34 packets respectively.



(a) ω_B =1.5 (b) ω_B =3

(c) ω_B =4 (d) ω_B =5

(e) ω_B =5.5 (f) ω_B =6

(g) ω_B =6.5 (h) ω_B =7

(i) ω_B =8 (j) ω_B =9

Fig. 4.6: Instantaneous Queue Size of the SU Controllers with W_U =10^5, A= 10^{-5} and M= 1.5 but Different ω_B Values

Table 4.6: Performance of SU Controllers with Different ω_B

W_U=10^5, A=10^{-5}, M=1.5	T_r (Second)	Peak Value (Packet)	μ_q (Packet)	σ_q(Packet)
ω_B =1.5	14	384	203	40
ω_B =3	13	326	202	39
ω_B =4	13	358	201	37
ω_B =5	8	296	202	33
ω_B =5.5	6	309	202	33
ω_B =6	7	288	205	33
ω_B =6.5	7	317	201	32
ω_B =7	7	311	207	31
ω_B =8	12	302	206	29
ω_B =9	14	309	203	34

Table 4.6 gives different performance measures of T_r , peak value, μ_q and σ_q as a function of ω_B while fixing W_U, M and A at 10^5, 1.5 and 10^{-5}, respectively. One can see that these measures are not monotonically decreasing w.r.t. ω_B. The smallest T_r of 6 to 8 seconds are achieved when ω_B is between 5 and 7. The ω_B =6 case has the smallest peak value (288 packets), followed by the ω_B=5.5 case (309 packers). Of these two cases, the ω_B=5.5 case has a smaller T_r, μ_q and σ_q. Therefore, we choose ω_B = 5.5 for our final SU controllers.

According to the simulation results mentioned above, we shall choose M=2, A=10^{-7} and ω_B=5.5 and therefore $W_s = \frac{s+11}{2s+11\times10^{-7}}$ to design the *SU* controller.

4.2.2.2 Effect of Varying W_u

Unlike the previous subsections, we now fix $W_s(s)$, and vary the weight function W_u to discover the effect of W_u on the SU controller performance. As discussed in Chapter 3, W_u is used to bound the input sensitivity function *U(s).* In our TCP/AQM system, W_u is specifically used to bound the variation of packet drop probability δ_p around the operating point p_0 =$2N^2/(R_0\times C)^2$ [HoMi01]. In addition, the variation δ_p should be negligible when compared to p_0 in order to have a stable system. Therefore, δ_p can be chosen 100 times (or more) smaller than p_0. Hence, W_u should be chosen 100 times (or more) greater than $1/p_0$. Therefore, when N=100 and R_0=

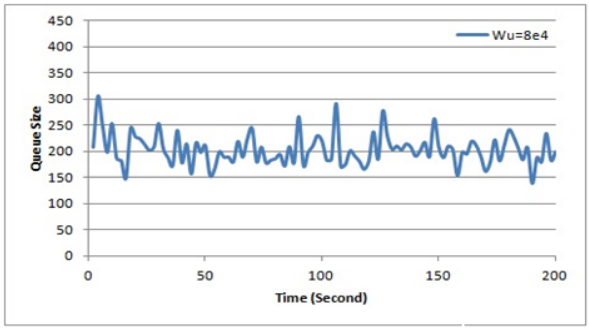
0.25 and C= 9708, p_0 = 3.395×10^{-3}, and 1/p_0=2.94×10^2. Hence, W_U can be chosen in the order of 10^4 or more. In the following study, we shall fix W_s with M=1.5, A=10^{-5} and ω_B=6 and derive different SU controllers corresponding to different values of W_u.

Table 4.7: Different SU Controllers Specification by Varying W_u

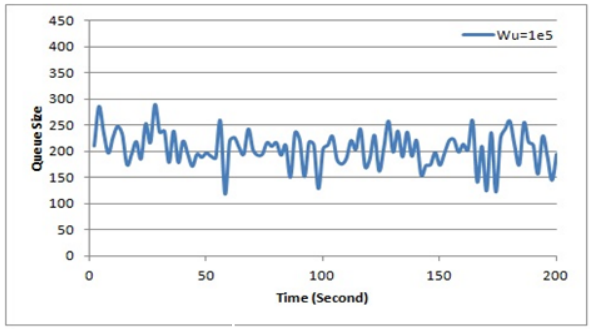
M=1.5, A =10^5, ω_B =6	γ_{min}	$\|E\|_\infty$	$\|E_2\|_2$	$e(\infty)$	PM, GM
W_U=8×10^4	6.4100	5.4769	3.3304	3.2158×10-5	57.2°, 7.7dB
W_U=10^5	6.7935	5.7828	3.5653	4.1416×10^{-5}	57.1°, 8.2dB
W_U=2×10^5	8.2054	6.9029	4.4359	4.7918×10^{-5}	58.2°, 9.0dB
W_U=5×10^5	10.7289	8.8889	6.0079	7.5693×10^{-5}	59.0°,10.9dB
W_U=10^6	13.3267	10.9037	7.6622	9.4731×10^{-5}	60.3 °, 12.2dB
W_U=10^7	31.5008	22.9520	21.5756	2.2459×10^{-4}	66.0°, 18.4dB

Table 4.7 shows γ_{min}, $\|\mathbf{E}_\infty\|_\infty$, $\|\mathbf{E_2}\|_2$, steady state error, e(∞), phase margin PM, and gain margin GM for different choices of W_U. One would observe that the measure γ_{min} becomes larger with increasing W_U meaning decreasing robustness. However, the steady state error e(∞) also increases especially for W_u=10^7 where e(∞) jumps to 2.2459x10^{-4}. Note that all the controllers are stable due to their positive PM and GM values. Table 4.7 suggests that one should pick a W_u as small as possible. However, there is the interplay of other parameters, which we shall examine next.

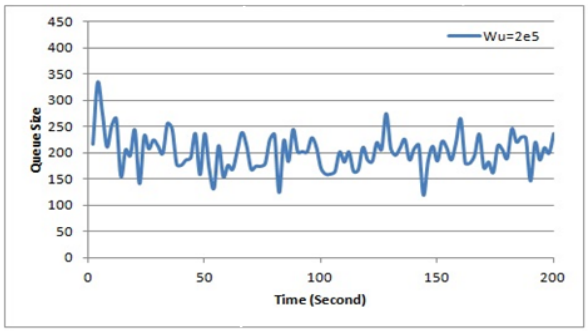
Fig. 4.7 and Table 4.8 show the performance measures of the rise/fall time T_r , the queue size peak value, the mean μ_q and the standard deviation σ_q of the queue length under different W_u. The smallest value of the rise/fall time T_r is 6 when W_u = 8×10^4 or 10^5. In addition, by increasing W_u, the rise/fall time increases until W_u reaches 10^7 then it drops to 37 seconds for W_u =10^7. Compared with other W_u values in Table 4.8, W_u= 8×10^4 and 10^5 have better performance measures of T_r (the smallest), the packet size peak values and queue variations while the mean queue length is slightly higher. We further choose W_u=10^5 for its smaller queue size peak value. It is worth mentioning that regarding to our simulation results, there is zero queue length for W_u less than 8×10^4.



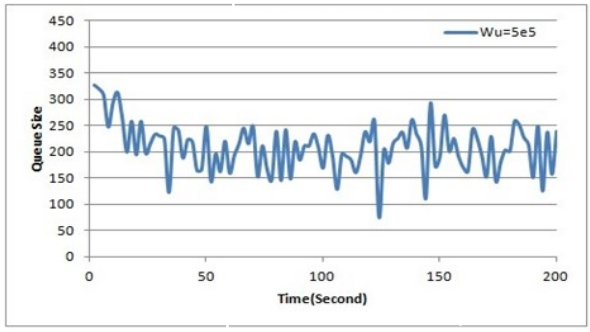
(a) W_u=8×10^4



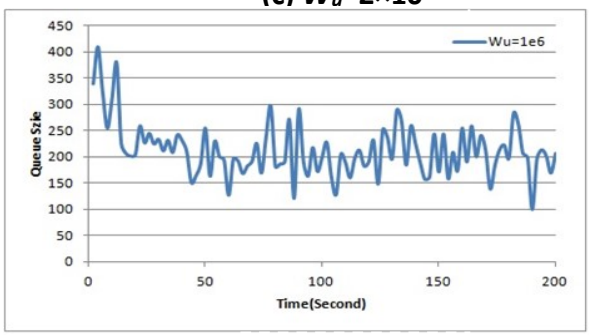
(b) W_u=10^5



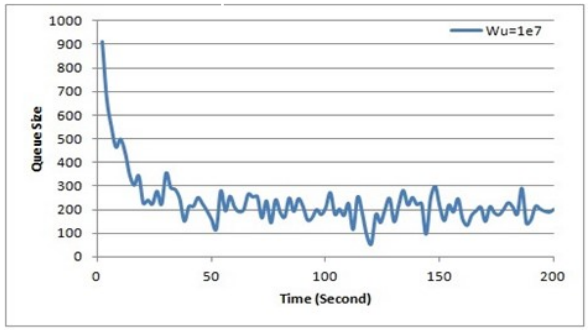
(c) W_u=2×10^5



(d) W_u=5×10^5



(e) W_u=10^6



(f) W_u=10^7

Fig.4.7: Instantaneous Queue Size of *SU* Controllers with A =10^5, ω_B= 6, M= 1.5 and Different W_u Values

Table 4.8: Performance of SU Controllers with Different W_u

M=1.5, A =10^{-5}, ω_B =6	T_r (Second)	Peak Value (Packet)	μ_q (Packet)	σ_q(Packet)
W_U=8×10^4	6	306	203	29
W_U=10^5	6	290	205	33
W_U=2×10^5	12	335	202	34
W_U=5×10^5	15	326	208	45
W_U=10^6	42	409	213	50
W_U=10^7	37	907	232	12

In view of the above study and many others, we have determined that M=2, A=$10^{-7,}$ ω_B=5.5 and W_u =10^5 are the parameter values that would give us an optimized SU controller. Readers may refer to Appendix D for more experiments using different W_s and W_u.

4.2.3 Stability Analysis

We shall now show the stability of the proposed SU controller for the TCP/AQM network which is an important traffic control system. Based on the system plant and the controller, we derive the general plant *G(s)* in the state pace domain using the Lyapunov's notation. We shall find the eigenvalues of the system to determine the existence of stability. For notation simplicity, we omit the time *t* in the following derivations.

To extend the original plant *P(s)* to the general plant model *G(s)* that would also include the weight functions, we first introduce the shorthand notation $\left(\frac{A}{C}\middle|\frac{B}{D}\right)$ to represent the transfer function $\boldsymbol{C}(s\boldsymbol{I}-\boldsymbol{A})^{-1}+\boldsymbol{D}$ for the standard state-space representation in time domain given by

$$\begin{cases}\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} + \boldsymbol{B}\boldsymbol{u} \\ \boldsymbol{y} = \boldsymbol{C}\boldsymbol{x} + \boldsymbol{D}\boldsymbol{u}\end{cases} \tag{4.6}$$

where $\boldsymbol{x}$ is state vector of the system; $\boldsymbol{u}$ is the input vector and $\boldsymbol{y}$ is the output vector of the system.

One can readily see that the sensitivity weight function $W_s(s)$ from eqn. 3.6 can be compactly written as follows

$$W_S(s) = \left(\frac{-\omega_b A}{\omega_b}\middle|\frac{1}{1/M}\right) \tag{4.7}$$

This can be construed as a transfer function for the state-space representation with A_{W_s}, B_{W_s}, C_{W_s} and D_{W_s}in a scalar form. This property shall be explored further in the stability proof of the controller later. Similarly, we come up with

$$W_U(s) = \left(\frac{A_{W_U}}{C_{W_u}}\middle|\frac{B_{W_U}}{D_{W_U}}\right) = \left(\frac{0}{0}\middle|\frac{0}{W_U}\right) \tag{4.8}$$

for an equivalent state-space representation in the time-domain.

4.2.3.1 Sate-Space Representation

We now proceed to the state-space formulation/derivation of our feedback control system.

With reference to Figs. 4.1 and 4.2, the weighted drop probability variation $\boldsymbol{\delta_P}$ is the input to the plant which is also the exogenous output $\boldsymbol{z_1}$, and the weighted queue size error $\boldsymbol{\delta_q}$ (queue size variation) is the exogenous output $\boldsymbol{z_3}$ in Fig. 4.2. Indeed, the queue size error is also the controller input v in Fig. 4.2 and the drop probability variation is the control signal $\boldsymbol{u}$. In addition, y_{ref} indicates reference (desire) queue size in the TCP/AQM system. Let $\boldsymbol{x_P}$ be the state vector. Then, P(s) can be written in the following state-space representation

$$\begin{cases} \dot{\boldsymbol{x_P}} = \boldsymbol{A_P x_P} + \boldsymbol{B_P \delta_p} \\ \boldsymbol{\delta_q} = \boldsymbol{C_P x_P} + \boldsymbol{D_P \delta_p} \end{cases}. \tag{4.9}$$

where $\boldsymbol{A_P}$ is the state matrix; $\boldsymbol{B_P}$ is the input matrix; $\boldsymbol{C_P}$ and $\boldsymbol{D_P}$ are the matrices associated with the output $\boldsymbol{\delta_q}$. Of the various forms of state-space representation, we shall use the Phase Variable Canonical form [Ogat02] derived as follows.

$$\begin{bmatrix} \dot{x}_{p_1} \\ \dot{x}_{p_2} \\ \dot{x}_{p_3} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -\frac{4N}{R_0^4 C} & -\left(\frac{6N}{R_0^3 C} + \frac{2}{R_0^2}\right) & -\left(\frac{2N}{R_0^2 C} + \frac{3}{R_0}\right) \end{bmatrix} \begin{bmatrix} x_{p_1} \\ x_{p_2} \\ x_{p_3} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \delta_p \tag{4.10}$$

$$\delta_q = \begin{bmatrix} \frac{C^2}{R_0 N} & \frac{-C^2}{2N} & 0 \end{bmatrix} \boldsymbol{x_p}$$

To represent the generalized plant G(s) in Fig. 4.2, note that the input now has two components: the disturbance $\boldsymbol{w}$ in addition to the normal input (or control purpose) $\boldsymbol{u}$. Let **x** be the state vector and $\boldsymbol{A_x}$ be the coefficient matrix associated with the generalized plan, G(s) along with the coefficient matrices $\boldsymbol{B_1}$ and $\boldsymbol{B_2}$ to be associated with the vectors $\boldsymbol{w}$ and $\boldsymbol{u}$ respectively. Note that in addition to the normal output $\boldsymbol{y}$, we also need to account for two additional outputs: $\boldsymbol{Z_\infty}$ for the H_∞ performance and $\boldsymbol{Z_2}$ for the H_2 performance. All of these outputs are general functions of the state vector $\boldsymbol{x}$ and the inputs of $\boldsymbol{w}$ and $\boldsymbol{u}$. So we shall let $\boldsymbol{C_y}$, $\boldsymbol{D_{y1}}$ and $\boldsymbol{D_{y2}}$ be the coefficient matrices associated with the vectors $\boldsymbol{x}$, $\boldsymbol{w}$ and $\boldsymbol{u}$ for the output $\boldsymbol{y}$; the coefficient matrices $\boldsymbol{C_2}$, $\boldsymbol{D_{21}}$, and $\boldsymbol{D_{22}}$, for the output $\boldsymbol{Z_2}$; and the coefficient matrices $\boldsymbol{C_\infty}$, $\boldsymbol{D_{\infty 1}}$ and $\boldsymbol{D_{\infty 2}}$, for the output $\boldsymbol{Z_\infty}$. Then the state space equation for the generalized plant can be written as

$$\begin{cases} \dot{\boldsymbol{x}} = \boldsymbol{A_x x} + \boldsymbol{B_1 w} + \boldsymbol{B_2 u} \\ \boldsymbol{Z_\infty} = \boldsymbol{C_\infty x} + \boldsymbol{D_{\infty 1} w} + \boldsymbol{D_{\infty 2} u} \\ \boldsymbol{Z_2} = \boldsymbol{C_2 x} + \boldsymbol{D_{21} w} + \boldsymbol{D_{22} u} \\ \boldsymbol{y} = \boldsymbol{C_y x} + \boldsymbol{D_{y1} w} + \boldsymbol{D_{y2} u} \end{cases} \tag{4.11}$$

To provide a state-space representation of the controller *K(s)* in Fig. 4.2, we let x_K be controller state vector. Then in the standard (*A, B, C, D*) notation, the controller *K(s)* alone can be

described by

$$\begin{cases} \dot{\boldsymbol{x}}_{\boldsymbol{K}} = \boldsymbol{A}_{\boldsymbol{K}}\boldsymbol{x}_{\boldsymbol{K}} + \boldsymbol{B}_{\boldsymbol{K}}\boldsymbol{y} \\ \boldsymbol{u} = \boldsymbol{C}_{\boldsymbol{K}}\boldsymbol{x}_{\boldsymbol{K}} + \boldsymbol{D}_{\boldsymbol{K}}\boldsymbol{y} \end{cases} \tag{4.12}$$

Finally, the closed-loop transfer function in Fig. 4.2 can also be defined and represented in the sate-space form. We let $\boldsymbol{x_{cl}}$ be closed-loop state vector, $\boldsymbol{A_{cl}}$ be the state matrix, $\boldsymbol{B_{cl}}$ be the input matrix associated with w; and $\boldsymbol{C_{cl1}}$ and $\boldsymbol{D_{cl1}}$ be the coefficient matrices associated with the vectors $\boldsymbol{x_{cl}}$ and w for the output $\boldsymbol{Z_\infty}$; and the coefficient matrices $\boldsymbol{C_{cl2}}$ and $\boldsymbol{D_{cl2}}$ for the output $\boldsymbol{Z_2}$. Denote $\boldsymbol{w}$ is the uncontrolled traffic input. Then we have

$$\begin{cases} \dot{\boldsymbol{x}}_{\boldsymbol{cl}} = \boldsymbol{A}_{\boldsymbol{cl}}\boldsymbol{x}_{\boldsymbol{cl}} + \boldsymbol{B}_{\boldsymbol{cl}}\boldsymbol{w} \\ \boldsymbol{Z}_\infty = \boldsymbol{C}_{\boldsymbol{cl1}}\boldsymbol{x}_{\boldsymbol{cl}} + \boldsymbol{D}_{\boldsymbol{cl1}}\boldsymbol{w} \\ \boldsymbol{Z}_{\boldsymbol{2}} = \boldsymbol{C}_{\boldsymbol{cl2}}\boldsymbol{x}_{\boldsymbol{cl}} + \boldsymbol{D}_{\boldsymbol{cl2}}\boldsymbol{w} \end{cases} \tag{4.13}$$

where it can be shown that [Sche95]

$$\boldsymbol{A}_{\boldsymbol{cl}} = \begin{pmatrix} \boldsymbol{A}_{\boldsymbol{x}} + \boldsymbol{B}_{\boldsymbol{2}}\boldsymbol{D}_{\boldsymbol{K}}\boldsymbol{C}_{\boldsymbol{y}} & \boldsymbol{B}_{\boldsymbol{2}}\boldsymbol{C}_{\boldsymbol{K}} \\ \boldsymbol{B}_{\boldsymbol{K}}\boldsymbol{C}_{\boldsymbol{y}} & \boldsymbol{A}_{\boldsymbol{K}} \end{pmatrix}, \tag{4.14}$$

$$\boldsymbol{B}_{\boldsymbol{cl}} = \begin{pmatrix} \boldsymbol{B}_{\boldsymbol{1}} + \boldsymbol{B}_{\boldsymbol{2}}\boldsymbol{D}_{\boldsymbol{K}}\boldsymbol{D}_{\boldsymbol{y1}} \\ \boldsymbol{B}_{\boldsymbol{K}}\boldsymbol{D}_{\boldsymbol{y1}} \end{pmatrix}$$

$$\boldsymbol{C}_{\boldsymbol{cl}} = \begin{pmatrix} \boldsymbol{C}_{\boldsymbol{cl1}} \\ \boldsymbol{C}_{\boldsymbol{cl2}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{C}_\infty + \boldsymbol{D}_{\infty\boldsymbol{2}}\boldsymbol{D}_{\boldsymbol{K}} & \boldsymbol{D}_{\infty\boldsymbol{2}}\boldsymbol{C}_{\boldsymbol{K}} \\ \boldsymbol{C}_{\boldsymbol{2}} + \boldsymbol{D}_{\boldsymbol{22}}\boldsymbol{D}_{\boldsymbol{K}} & \boldsymbol{D}_{\boldsymbol{22}}\boldsymbol{C}_{\boldsymbol{K}} \end{pmatrix}, \text{ and } \boldsymbol{D}_{\boldsymbol{cl}} = \begin{pmatrix} \boldsymbol{D}_{\boldsymbol{cl1}} \\ \boldsymbol{D}_{\boldsymbol{cl2}} \end{pmatrix} = \begin{pmatrix} \boldsymbol{D}_{\infty\boldsymbol{1}} + \boldsymbol{D}_{\infty\boldsymbol{1}}\boldsymbol{D}_{\boldsymbol{K}}\boldsymbol{D}_{\boldsymbol{y2}} \\ \boldsymbol{D}_{\boldsymbol{21}} + \boldsymbol{D}_{\boldsymbol{21}}\boldsymbol{D}_{\boldsymbol{K}}\boldsymbol{D}_{\boldsymbol{y2}} \end{pmatrix},$$

and $\boldsymbol{A_x}$, $\boldsymbol{B_2}$ and $\boldsymbol{C_y}$ are matrices of the general H_2/H_∞ controller [Mack04, Matl13] arising from our plant model *P(s).* We use the Robust Control-toolbox in MATLAB [Matl13] to determine how the states are interconnected and therefore to determine each element in the state-space matrix of G(s) as a function of the matrices of weight functions and the original plant model. Through a bit of reverse engineering using prime numbers (to identify which sub-matrices are involved in the multiplication process), it is determined that

$$\boldsymbol{A}_{\boldsymbol{x}} = \begin{bmatrix} [\boldsymbol{A}_{\boldsymbol{P}}]_{3\times3} & [0]_{3\times1} \\ -\boldsymbol{C}_{\boldsymbol{P}}\boldsymbol{B}_{\boldsymbol{W_S}} & \boldsymbol{A}_{\boldsymbol{W_S}} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ -\dfrac{4N}{R_0^4 C} & -\left(\dfrac{6N}{R_0^3 C} + \dfrac{2}{R_0^2}\right) & -\left(\dfrac{2N}{R_0^2 C} + \dfrac{3}{R_0}\right) & 0 \\ -\dfrac{C^2}{R_0 N} & \dfrac{C^2}{2N} & 0 & -w_b A \end{bmatrix}$$

$$B_2 = \begin{bmatrix} [\boldsymbol{B}_{\boldsymbol{P}}]_{3\times1} \\ -\boldsymbol{B}_{\boldsymbol{W_S}}\boldsymbol{D}_{\boldsymbol{P}} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$C_y = [[-\boldsymbol{C_P}]_{1\times 3} \quad 0] = \begin{bmatrix} \frac{C^2}{R_0 N} & \frac{-C^2}{2N} & 0 & 0 \end{bmatrix} \tag{4.15}$$

Therefore, by using eqn. (4.14) $\boldsymbol{A_{cl}}$ can be presented by

$$\boldsymbol{A_{cl}} = \begin{bmatrix} \boldsymbol{A_x} + \boldsymbol{B_2 D_K C_y} & \boldsymbol{B_2 C_K} \\ \boldsymbol{B_K C_y} & \boldsymbol{A_K} \end{bmatrix} =$$

$$\begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ -\frac{4N}{R_0^4 C} + D_k \frac{C^2}{R_0 N} & -\left(\frac{6N}{R_0^3 C} + \frac{2}{R_0^2}\right) - D_k \frac{C^2}{2N} & -(\frac{2N}{R_0^2 C} + \frac{3}{R_0}) & 0 & C_{K_1} & C_{K_2} & C_{K_3} & C_{K_4} \\ -\frac{C^2}{R_0 N} & \frac{C^2}{2N} & 0 & -w_b A & 0 & 0 & 0 & 0 \\ B_{K_1} \frac{C^2}{R_0 N} & B_{K_1} \frac{-C^2}{2N} & 0 & 0 & A_{K_{11}} & A_{K_{12}} & A_{K_{13}} & A_{K_{14}} \\ B_{K_2} \frac{C^2}{R_0 N} & B_{K_2} \frac{-C^2}{2N} & 0 & 0 & A_{K_{21}} & A_{K_{22}} & A_{K_{23}} & A_{K_{24}} \\ B_{K_3} \frac{C^2}{R_0 N} & B_{K_3} \frac{-C^2}{2N} & 0 & 0 & A_{K_{31}} & A_{K_{32}} & A_{K_{33}} & A_{K_{34}} \\ B_{K_4} \frac{C^2}{R_0 N} & B_{K_4} \frac{-C^2}{2N} & 0 & 0 & A_{K_{41}} & A_{K_{42}} & A_{K_{43}} & A_{K_{44}} \end{bmatrix} \tag{4.16}$$

where $A_{K_{ij}}$ is the (*i, j*) entry of the matrix $\boldsymbol{A_K}$ in eqn. (4.12) B_{K_i} is the i^{th} element of column vector $\boldsymbol{B_K}$ in eqn. (4.12); and C_{K_i} the i^{th} element of row vector $\boldsymbol{C_K}$ in eqn. (4.12).

4.2.3.2 Framework of the Stability Proof

We shall use the Lyapunov's First Method (also called by the Indirect Method) to check/test the stability of our controller. Since this method is used to study the internal stability of a non-linear system under zero-input response (i.e., the response of $\dot{\boldsymbol{x}} = \boldsymbol{A}(\boldsymbol{t})\boldsymbol{x}$), it can be readily used for a linear system as a special case where the state-space representation parameters are constant.

Theorem 4.1: Consider a linear system with the state space model

$$\dot{\boldsymbol{x}} = \boldsymbol{A}\boldsymbol{x} \tag{4.17}$$

where $\boldsymbol{A}$ is a finite matrix. The system is asymptotically stable (in fact, exponentially stable) if all real parts of the eigenvalues of $\boldsymbol{A}$ are negative. Except for the boundary situation, the eigenvalues of the linearized version can be used to reveal the stability properties of an equilibrium point of a nonlinear system [BaRo06] as follows.

1) The system is asymptotically stable if $Re(\lambda_i)<0$ for all eigenvalues of $\boldsymbol{A}$.

2) The system is unstable if $Re(\lambda_i)>0$ for one or more eigenvalues of $\boldsymbol{A}$.

3) If $Re(\lambda_i) \leq 0$ for all i and at least one $Re(\lambda_i)=0$, then $\boldsymbol{x} = 0$ may be either stable, asymptotically stable or unstable for the nonlinear system [BaRo06].

We can see that the above theorem is applicable to our closed-loop system of eqn. (4.13) which relies on the linear systems with state-space realization like eqn. (4.17) by considering the zero-input response. Therefore, we can apply Theorem 4.2 [Vinn01] below to prove the stability.

Theorem 4.2: For the linear time invariant system, using Lyopunov's first method, the controller *K(s)* is called *stabilizing* if $\boldsymbol{A_{cl}}$ is exponentially stable, i.e $Re(\lambda_i)<0$ for all eigenvalues of $\boldsymbol{A_{cl}}$ where $\boldsymbol{A_{cl}}$ is the state matrix of the closed loop state-space realization.

In summary, one needs to take the following steps to prove the stability of the H_2/H_∞ controller:

1) Present the plant model in the state-space model.

2) Combine *P(s)* with the weight functions to give the generalized plant model *G(s).*

3) Rewrite the state-space representation of the *G(s)* as eqn. (4.11).

4) Rewrite the proposed controller in the state-space representation as eqn. (4.12)

5) Find $\boldsymbol{A_{cl}}$, the closed-loop state matrix according to eqn. (4.14) and (4.16)

6) Find the eigenvalues of $\boldsymbol{A_{cl}}$

7) See if the *K(s)* is stabilizing according to the conditions of Theorem 4.2.

Since there is no closed-form expression for *K(s)* that can only be arrived at by iteration, we have to resort to a numerical proof as our next best step.

4.2.3.3 Numerical Proof Examples

In this section, we analysis the stability of the designed controller in Section 4.2.2. Therefore, the system can be represented by a plant function $P(s)\frac{471226.32}{(s+0.3296)(s+4)}e^{-0.25s} =$ [HoMi02]. It can be represented in state space representation as following:

$$\begin{bmatrix} \dot{x_{p_1}} \\ \dot{x_{p_2}} \\ \dot{x_{p_3}} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ -10.548 & -35.9555 & -12.3296 \end{bmatrix} \begin{bmatrix} x_{p_1} \\ x_{p_2} \\ x_{p_3} \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \boldsymbol{u} \quad (4.18)$$

$$\boldsymbol{\delta_q} = [3769810.56 \quad -471226.32 \quad 0]\boldsymbol{x_p}$$

As mentioned earlier, the general plant function *G(s*) contains the original plant model *P(s),* and the two weight functions $W_S(s)$ and $W_U(s)$. After doing several simulations to find acceptable controller, mentioned in Section 4.2.2, W_S is chosen with M=2, A=10^{-7} and ω_b=5.5. These give us $W_s(s) = \frac{s+11}{2s+11\times10^{-7}}$. In addition, we have determined W_U =10^5. According to eqns. (4.7) and (4.8), they can be presented in state space form as following.

$$W_S(s) = \left(\frac{A_{W_S}}{C_{W_S}}\middle|\frac{B_{W_S}}{D_{W_S}}\right) = \left(\frac{-\omega_b A}{\omega_b}\middle|\frac{1}{1/M}\right) = \left(\frac{-5.5\times10^{-7}}{5.5}\middle|\frac{1}{0.5}\right) \quad (4.19)$$

$$W_U(s) = \left(\frac{A_{W_U}}{C_{W_U}}\middle|\frac{B_{W_U}}{D_{W_U}}\right) = \left(\frac{0}{0}\middle|\frac{0}{W_U}\right) = \left(\frac{0}{0}\middle|\frac{0}{10^5}\right) \quad (4.20)$$

For the designed SU controller in Section 4.2.1.1, we obtain the controller in the state-space representation as eqn. (4.12) as

$$K(s) = \left(\begin{array}{cccc|c} 0.0013218 & 0.0002618 & -0.000024 & -0.00000433 & -14.9049 \\ 44.08627 & -12.69549 & 1.90533 & -0.032206 & -75.7658370 \\ 113.42440 & -25.45972 & 2.607482 & -0.5767964 & -177.2341 \\ -1566.57545 & 286.87001 & -22.6383 & -7.80094 & 2428.1489 \\ \hline 0.0074043\times10^{-7} & 0.0895378\times10^{-7} & -0.010977\times10^{-7} & 0.3180273\times10^{-7} & 0 \end{array}\right) \quad (4.21)$$

Substituting eqns. (4.18-4.21) in eqn. (4.16), one would obtain

$$A_{cl} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ -10.548002 & -35.955501 & -12.329625 & 0 & 7.4043\times10^{-10} & 8.95378\times10^{-9} & -1.0977\times10^{-9} & 3.180273\times10^{-8} \\ -8.84098\times10^{6} & 1.10512\times10^{6} & 0 & -1\times10^{-6} & 0 & 0 & 0 & 0 \\ 5.618\times10^{7} & -7.023\times10^{6} & 0 & 0 & -1.322\times10^{-3} & 2.62\times10^{-4} & -2.4\times10^{-5} & -4\times10^{-6} \\ 0.28562\times10^{9} & -0.035702\times10^{9} & 0 & 0 & 44.086271 & -12.695492 & 1.905339 & -3.2207\times10^{-2} \\ 0.66813\times10^{9} & -0.083517\times10^{9} & 0 & 0 & 113.424409 & -25.459724 & 2.607483 & -0.576796 \\ -9.15366\times10^{9} & 1.144207\times10^{9} & 0 & 0 & -1566.575452 & 286.870015 & -22.638305 & -7.800946 \end{bmatrix} \quad (4.22)$$

from which the eigenvalues are obtained using MATLAB as

$$\lambda = [-5.5\times10^{-10} \quad -0.4087 \quad -1.5291 + 2.269i \quad -1.5291 - 2.269i \quad -3.109 \quad -4.012 \quad -8.0002 \quad -11.63]$$

All these eigenvalues have negative real parts. Therefore, $\boldsymbol{A_{cl}}$ is exponentially stable, and the controller *K(s)* is *stabilizing* according to Theorem 4.2.

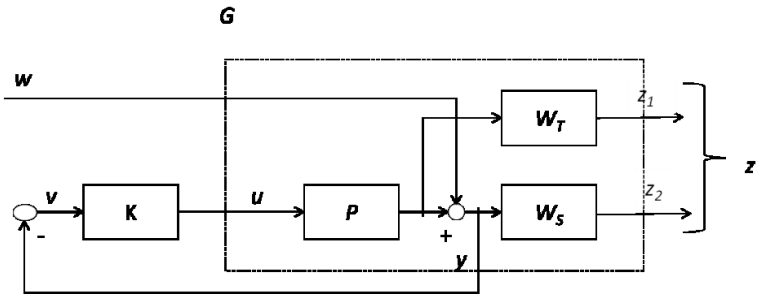


Fig. 4.8: General Configuration for ST Controller

4.3 The ST Controller

As a different type of H_2/H_∞ controller, the ST controller makes use of sensitivity function and complementary sensitivity function in Fig. 4.1. There is no limitation on Input sensitivity function in this controller by omitting $W_U(s)$ in Fig. 2.4. Therefore, the output vector is simply $\boldsymbol{z} = [z_1, z_2]$. By skipping the Input Sensitivity Function in the design procedure, the general plant function *G(s)* now contains only the original plant model *P(s)*, and the two weight functions $W_S(s)$ and $W_T(s)$.

4.3.1 Controller Design

Since there is no W_U in the ST controller design, its design would be similar to Fig. 3.1 except for Step-2 by not initializing W_U. Also, there is no need to check the condition $|U(j\omega)|<\gamma_{min}/|W_U|$ in Step-6. To find the generalized plant *G(s)* in Fig. 4.8, we need to specify the weight functions $W_S(s)$ and $W_T(s)$ based on the information given in Section 3.2. Similar to the SU controller, we shall use simulations to obtain their values. Section 4.3.3 provides a design example on the selection of these weight functions and its impacts on the performance of the TCP/AQM system, and hence illustrating how we can arrive at our choice.

4.3.1.1 Design Example

We shall show an example to design the ST controller for the TCP/AQM network using the same parameter values as the SU controller in Section 4.2.1.1. That is, *N*= 100 greedy ftp sources using TCP sessions and 20 http sources, round trip time R_0=0.25s, and the bandwidth of the

bottleneck link at *C*=9708 packets/second.

After doing simulations to find the acceptable controller like the SU controller, $W_T(s)$ is chosen with a_1= 0.00031, a_2= 3.5 and a_3=0.01. This gives us $W_T(s) = \frac{0.00031+3.5s}{1+0.01s}$. Note that we still have to obtain the function $W_S(s) = \frac{s+11}{2s+11\times10^{-7}}$ as found for the SU controller.

Similar to the SU controller, we use the *Robust Control*-toolbox in MATLAB [Matl13] to solve the iterative optimization design problem of eqn. (3.1). We do this by assigning the same weights of [α , β] =[1, 1] to $\|\boldsymbol{E}_\infty\|_\infty$ and $\|\boldsymbol{E}_2\|_2$, and [δ_0 , ν_0] = [200, 200]. The final ST controller is obtained as

$$K(s) = \frac{1.522\times10^{-10}s^4 - 2.508\times10^{-7}\,s^3 - 2.201\times10^{-6}\,s^2 - 2.54\times10^{-7}s + 1.706\times10^{-6}}{2.23\times10^{-20}s^5 + 3.457\times10^{-10}s^4 + 0.05637\,s^3 + 0.5711\,s^2 + 0.819\,s - 2.855\times10^{-6}}.$$

The H∞-norm of the closed-loop transfer functions from $\boldsymbol{w}$ to $\boldsymbol{Z_\infty}$ is $\|\boldsymbol{E}_\infty\|_\infty = 9.8$, and the H_2-norm of closed-loop transfer functions from $\boldsymbol{w}$ to $\boldsymbol{Z_2}$ is $\|\boldsymbol{E}_2\|_2 = 22.3$. These two norms give the minimum of the mixed H_2/H_∞ norm at $\gamma_{min} = 24.4$.

Through the Bode diagrams, one can show that our designed W_S and W_T are feasible functions as they meet the expected criteria mentioned in Section 3.2. Readers may refer to Appendix C for more information.

4.3.2 Effect of the Design Parameters

In this study, we use the same example as in Section 4.3.1.1 to investigate the impacts of the weight functions of W_T and W_S on the system performance. Specifically, we vary the parameters a_1, a_2 and a_3 (see eqn. 3.7) while fixing W_S. Then we vary the parameters ω_B, *M* or *A* *in* W_S while fixing W_T. Again, keep in mind that every change in a parameter value means that we have designed a new controller.

4.3.2.1 Effect of Varying W_T

We shall fix W_S at $W_S(s) = \frac{s+11}{2s+11\times10^{-7}}$ in this study and vary the parameters in $W_T(s) = \frac{a_1+a_2s}{1+a_3s}$. As discussed in Section 3.2, the approach is to have a_1 small to ensure that $|W_T|$ is small at low frequencies, while having $a_3 << a_2$ to guarantee that $|W_T|$ is large at high frequencies.

a) Varying a_1

In order to find the proper a_1 for a good controller, we set different values to a_1 while a_2 and a_3 are fixed at 3.5, 0.01.

Table 4.9: Different ST Controllers Specifications by Varying a_1

a_2=3.5, a_3=10^{-2}	γ_{min}	$\|\boldsymbol{E}_\infty\|_\infty$	$\|\boldsymbol{E}_2\|_2$	$e(\infty)$	PM, GM
a_1=0.0001	23.8759	10.2318	21.5724	1.1360×10^{-6}	62.9°, 14.7dB
a_1=0.00011	26.5506	12.3808	23.4872	1.5532×10^{-5}	69.2°, 19.4dB
a_1=0.0003	22.9752	9.2837	21.0160	3.1877×10^{-6}	60.0°,16.1dB
a_1=0.00031	24.4068	9.8043	22.3510	2.6424×10^{-6}	53.7°, 19.4dB
a_1=0.01	29.8718	14.3114	26.2203	1.4072×10^{-5}	40.2°,21.7dB
a_1=0.011	26.5821	12.8961	23.2443	2.8417×10^{-6}	56.3°, 16.2dB
a_1=0.012	27.6165	13.9255	23.8485	1.1750×10^{-5}	59.1°, 15.6dB

Table 4.9 compares the performance of different controllers' specifications (by varying a_1) in terms of their γ_{min} , $\|E\|_\infty$, $\|E_2\|_2$, $e(\infty)$, the phase margin PM, and the gain margin GM. No specific trend is observed as a_1 is increasing. In fact, there are no significant differences among their performance results except for $e(\infty)$ which can have difference in one order of magnitude. All the controllers are robust since γ_{min} , $\|\boldsymbol{E}_\infty\|_\infty$, $\|\boldsymbol{E}_2\|_2$ have finite values and they are stable because the PM (Phase Margin) and GM (Gain Margin) are positive. The three smallest values for γ_{min} , $\|\boldsymbol{E}_\infty\|_\infty$, $\|\boldsymbol{E}_2\|_2$ are obtained for a_1=0.0001, 0.0003. 0.000031.

Note that we have actually tried a_1 ranging from 0.0001 to 0.1, but not all of them are stable and robust for use with the TCP/AQM system. For example, we found that a_1=0.1, 0.007 and 0.0001 are unstable and a_1=0.001 and 0.005 are not robust.

Fig. 4.9 shows the OPNET simulation of the instantaneous queue size for the ST controller. One can see that a_1=0.01 has significantly larger peak queue size value than other a_1 values. However, the peak queue size is almost same for other a_1 values. All the results show that the controllers are stable and robust and are able to reach to desired queue size value.

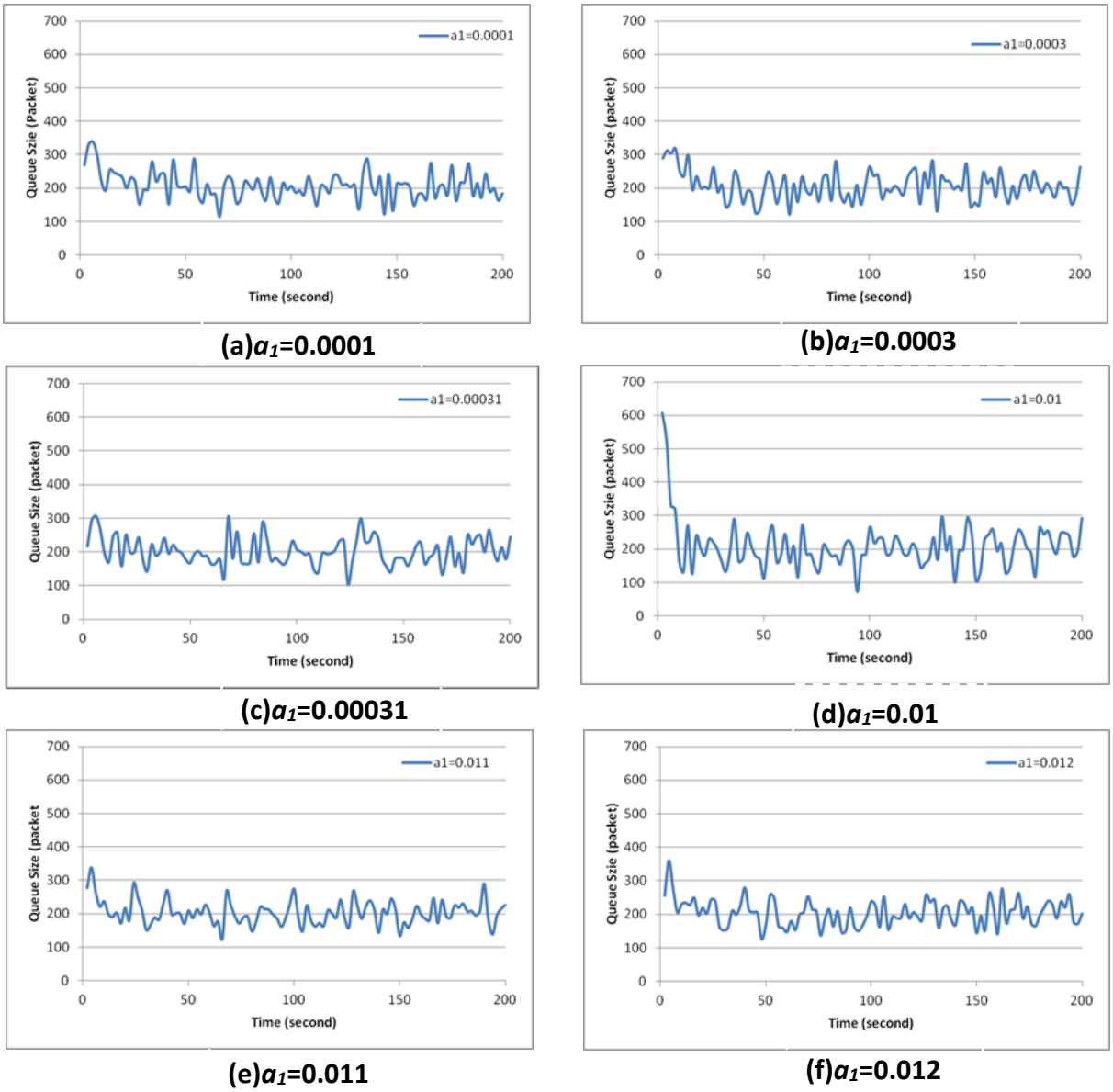


Fig. 4.9: Instantaneous Queue Size of ST Controllers with a_2=3.5 and a_3=0.01 but Different a_1 Values

Table 4.10 tabulates and compares the rise/fall time T_r, the peak value, the mean queue length μ_q, and the standard deviation σ_q extracted from the experiments in Fig. 4.9. One can see that T_r, peak value, μ_q and σ_q are not monotonically decreasing w.r.t. a_1. The best results with the smallest T_r are achieved for a_1=0.01, 0.001. Between these values of a_1, the smallest μ_q, σ_q hence less deviation from the mean is obtained at a_1=0.00031. Note that the performance results are sensitive to small changes in a_1. For example, a_1=0.0003 and a_1=0.00031 have very

different T_r value of 15 and 9 seconds respectively although they are close values. According to all discussions above, we choose a_1=0.00031 for our final ST controllers.

Table 4.10: Performance of ST Controllers with Different a_1

a_2=3.5, a_3=10^{-2}	T_r (Second)	Peak Value (Packet)	μ_q (Packet)	σ_q(Packet)
a_1=0.0001	10	326	208	41
a_1=0.00011	12	437	211	51
a_1=0.0003	15	318	207	42
a_1=0.00031	9	306	201	41
a_1=0.01	9	608	209	71
a_1=0.011	12	308	204	36
a_1=0.012	17	358	204	39

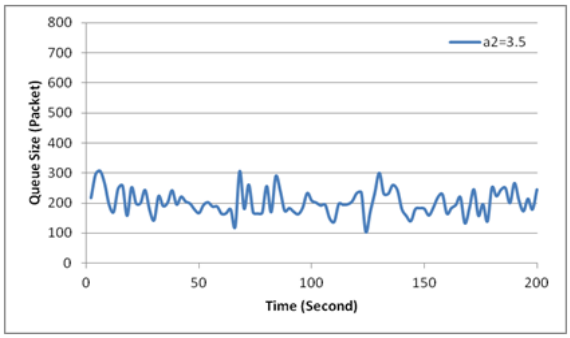
b) Varying a_2

We shall investigate the impact of various a_2 with fixed a_1 and a_3 at a_1=0.001, a_3 = 0.01, and $W_s = \frac{s+11}{2s+11\times10^{-7}}$.

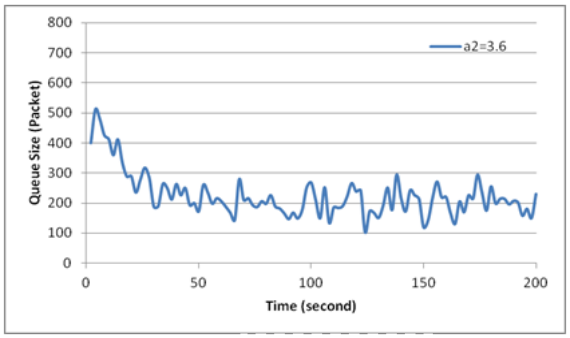
Table 4.11: Different ST Controllers Specifications by Varying a_2

a_1=0.00031, a_3=10^{-2}	γ_{min}	$\|\boldsymbol{E}_\infty\|_\infty$	$\|\boldsymbol{E}_2\|_2$	$e(\infty)$	PM, GM
a_2=3.5	24.4068	9.8043	22.3510	2.6424×10^{-6}	53.7°,19.4dB
a_2=3.6	36.0715	20.6942	29.5450	2.0598×10^{-6}	89.3°,21.4dB
a_2=4	40.0019	24.1991	31.8521	2.1223×10^{-6}	71.5°,29.3dB
a_2=4.2	39.2891	22.7018	32.0665	3.8340×10^{-6}	67.0°,11.3dB

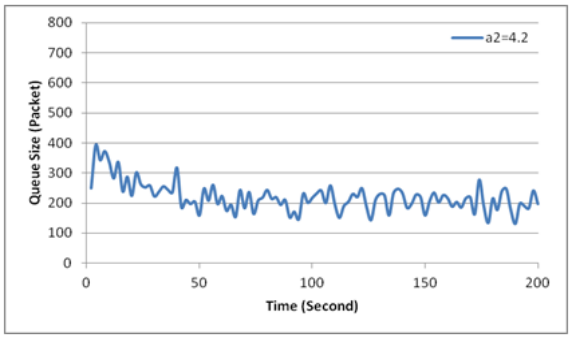
Table 4.11 summarizes the controllers specifications in terms of γ_{min}, $\|\boldsymbol{E}_\infty\|_\infty$, $\|\boldsymbol{E}_2\|_2$, $e(\infty)$, the PM, and the GM, all as a function of a_2. The table shows that all the controllers are robust regarding to have finite γ_{min} values and are stable because the PM and GM are positive. One can see that by increasing a_2, except for a_2=4.2, $\|\boldsymbol{E}_\infty\|_\infty$, $\|\boldsymbol{E_2}\|_2$ and γ_{min} all increase. However, the smallest value for $\|\boldsymbol{E}_\infty\|_\infty$, $\|\boldsymbol{E_2}\|_2$ and γ_{min} is obtained at a_2=3.5.



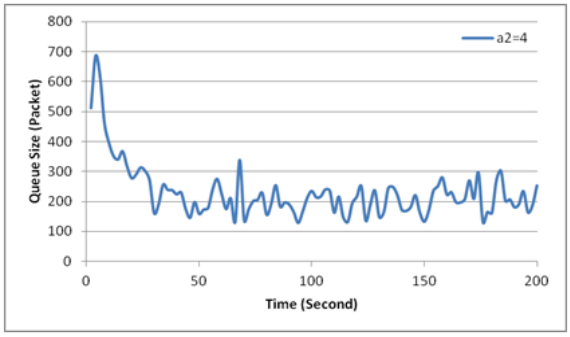
(a) a_2=3.5



(b) a_2=3.6



(c) a_2=4.2



(d) a_2=4

Fig. 4.10: Instantaneous Queue Size of ST Controllers with a_1=0.00031 and a_3=0.01 but Different a_2 Values

Table 4.12: Performance of ST Controllers with Different a_2

a_1=3.5, a_3=10^{-2}	T_r (Second)	Peak Value (Packet)	μ_q (Packet)	σ_q(Packet)
a_2=3.5	9	301	201	41
a_2=3.6	29	505	225	71
a_2=4	28	648	229	89
a_2=4.2	41	390	219	47

Fig 4.10 and Table 4.12 present the OPNET simulation results for the instantaneous queue size for varying a_2 while other parameters are fixed. The smallest value of the rise/fall time T_r is 9 second when a_2=3.5. Other T_r given in Table 4.12 are significantly larger than 9 sec (e.g. 29 sec for a_2=3.6 and 41 second for a_2=4.2). Therefore, we choose a_2=3.5 for our final ST controller because it gives the smallest peak queue size value (301 packets) along with μ_q = 201 packets and σ_q= 41.

Note that the ST controller is sensitive to different parameters of the $W_T(s)$. We have

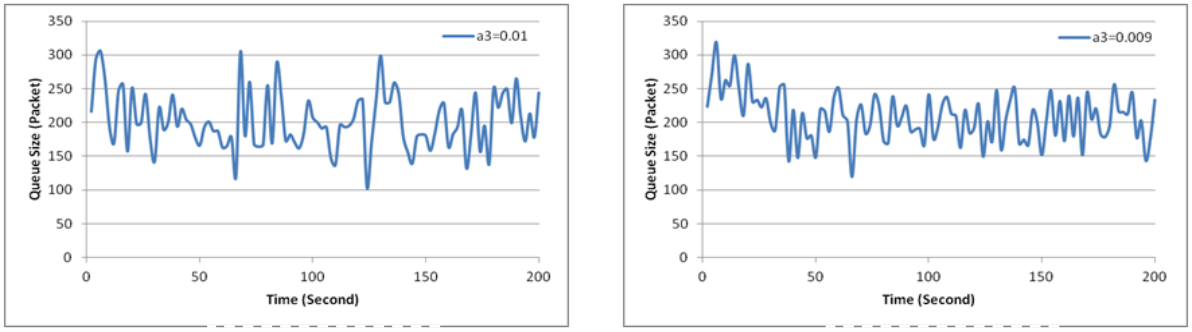
actually tried various a_2 values ranging from 0.01 to 10, but not all of them are stable and robust for the TCP/AQM system. For example, we found that a_2=2.4, 7 and 0.0001 give unstable systems and systems with a_2=2, 5 and 10 are not robust.

Table 4.13: Different ST Controllers Specification by Varying a_3

a_1=0.00031, a_2=3.5	γ_{min}	$\|\|\boldsymbol{E}_\infty\|\|_\infty$	$\|\|\boldsymbol{E}_2\|\|_2$	$e(\infty)$	PM, GM
a_3=0.01	24.4068	9.8043	22.3510	2.6424×10^{-6}	53.7°. 19.4dB
a_3=0.009	34.0078	18.4552	28.5646	3.1481×10^{-5}	55.5°, 9.4dB

c) Varying a_3

The impact of a_3 is now investigated by changing a_3 and fixing a_1 and a_2 at 0.000031 and 3.5 respectively. Similar to previous observations on other parameters, the ST controller is sensitive to different a_3 values. Through a large range of a_3, Table 4.13 presents the controller specification of the few a_3 values, which give both stable and robust controllers. It is clear from the table that by using a_3 = 0.01, smaller γ_{min}, $\|\boldsymbol{E}_\infty\|_\infty$, $\|\boldsymbol{E}_2\|_2$, $e(\infty)$ an PM and GM are achieved.



(a) a_3=0.01 **(b) a_3=0.009**

Fig. 4.11: Instantaneous Queue Size of the ST Controllers with a_1=0.00031 and a_2=3.5 but Different a_3 Values

Table 4.14: Performance of ST Controllers with Different a_3

a_1=0.00031, a_2=3.5	T_r (Second)	Peak Value (Packet)	μ_q (Packet)	σ_q(Packet)
a_3=0.01	9	301	201	41
a_3=0.009	41	310	225	78

One can see from Fig. 4.11 that both controllers are stable and robust since the results reaches to desired queue size, 200. It is apparent from Table 4.14 that a_3= 0.01 creates shorter rise/fall time (9 seconds for a_3= 0.01 to 41 seconds for a_3= 0.009). In addition, a_3= 0.01 yields smaller queue size peak value (301 packets to 310packets), μ_q (201 packets to 225 packets) and σ_q (41 packets to 78 packets). Therefore, a_3= 0.01 is chosen for final ST controller.

Note that it is difficult to obtain an operative a_3 parameter for the ST controller is we have actually tried a_3 ranging from 0.0001 to 0.1, but not all of them are stable and robust for use with the TCP/AQM system. For example, we found that a_3=0.05 is unstable and a_3=0.001 and 0.005 are not robust.

The following controller can be drawn from this section that the proper complementary sensitivity weight function is $W_T(s) = \frac{0.00031+3.5s}{1+0.01s}$.

4.3.2.2 Effect of Varying W_S

In this study, we fix the $W_T(s)$ while the effect of changing the $W_S(s)$ are observing. For the comfort of the readers and avoiding the repeating of the similar studies, readers may refer to the Appendix E for the details. Note that we have tried wide ranges of W_s parameters values. However, most of the parameters values cannot yield a valid/operative ST controller, which their queue size is zero all the time. As mentioned before in various places of Section 4.3.2.1, the ST controller is sensitive to the weight functions parameters values and it is difficult to find the values to obtain a valid ST controller with satisfactory performance

4.3.3 Stability Analysis

We shall provide the stability analysis of the proposed ST controller in the state space domain using Lyapunov's Indirect Method. The same procedure to SU controller will be applied here. The only difference is that the general plant model $G(s)$ now consists of $W_s(s)$, $W_T(s)$ and $P(s)$. Similar to the state-space representation of the $W_s(s)$ mentioned in eqn. (4.7), we can come up with

$$W_T(s) = \left(\frac{A_{W_T}}{C_{W_T}}\middle|\frac{B_{W_T}}{D_{W_T}}\right) = \left(\frac{-1/a_3}{a_2/a_3}\middle|\frac{1}{a_1/a_3}\right) \quad (4.23)$$

4.3.3.1 State-Space Model

Since the ST controller uses the complementary sensitivity function $W_T(s)$, instead of the input sensitivity function in the SU controller, the generalized plant model, $\boldsymbol{A_x}$, $\boldsymbol{B_2}$, $\boldsymbol{C_y}$ that are associated to the general plant model, and hence the closed-loop state matrix $\boldsymbol{A_{cl}}$ is different from eqns. (4.15) and (4.16). Similar to procedure in Section 4.2.3.1, $\boldsymbol{A_x}$, $\boldsymbol{B_2}$ and $\boldsymbol{C_y}$ can be written as follows.

$$A_x = \begin{bmatrix} [A_P]_{3\times3} & [0]_{3\times2} \\ \begin{bmatrix} -C_P B_{W_s} \\ -C_P B_{W_T} \end{bmatrix} & \begin{bmatrix} A_{W_s} & 0 \\ 0 & A_{W_T} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ -\frac{4N}{R_0^4 C} & -\left(\frac{6N}{R_0^3 C} + \frac{2}{R_0^2}\right) & -(\frac{2N}{R_0^2 C} + \frac{3}{R_0}) & 0 & 0 \\ -\frac{C^2}{R_0 N} & \frac{C^2}{2N} & 0 & -w_b A & 0 \\ -\frac{C^2}{R_0 N} & \frac{C^2}{2N} & 0 & 0 & -\frac{1}{a_3} \end{bmatrix}$$

$$B_2 = \begin{bmatrix} [B_P]_{3\times1} \\ -B_{W_s} D_P \\ B_{W_T} D_P \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$C_y = [[-C_P]_{1\times3} \quad 0 \quad 0] = [\frac{C^2}{R_0 N} \quad \frac{-C^2}{2N} \quad 0 \quad 0 \quad 0] \qquad (4.24)$$

By substituting in eqn. (4.14), one would obtain

$$A_{cl} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{4N}{R_0^4 C} + D_k \frac{C^2}{R_0 N} & -\left(\frac{6N}{R_0^3 C} + \frac{2}{R_0^2}\right) - D_k \frac{C^2}{2N} & -(\frac{2N}{R_0^2 C} + \frac{3}{R_0}) & 0 & 0 & C_{K_1} & C_{K_2} & C_{K_3} & C_{K_4} & C_{K_5} \\ -\frac{C^2}{R_0 N} & \frac{C^2}{2N} & 0 & -w_b A & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{C^2}{R_0 N} & \frac{C^2}{2N} & 0 & 0 & -\frac{1}{a_3} & 0 & 0 & 0 & 0 & 0 \\ B_{K_1} \frac{C^2}{R_0 N} & B_{K_1} \frac{-C^2}{2N} & 0 & 0 & 0 & A_{K_{11}} & A_{K_{12}} & A_{K_{13}} & A_{K_{14}} & A_{K_{15}} \\ B_{K_2} \frac{C^2}{R_0 N} & B_{K_2} \frac{-C^2}{2N} & 0 & 0 & 0 & A_{K_{21}} & A_{K_{22}} & A_{K_{23}} & A_{K_{24}} & A_{K_{25}} \\ B_{K_3} \frac{C^2}{R_0 N} & B_{K_3} \frac{-C^2}{2N} & 0 & 0 & 0 & A_{K_{31}} & A_{K_{32}} & A_{K_{33}} & A_{K_{34}} & A_{K_{35}} \\ B_{K_4} \frac{C^2}{R_0 N} & B_{K_4} \frac{-C^2}{2N} & 0 & 0 & 0 & A_{K_{41}} & A_{K_{42}} & A_{K_{43}} & A_{K_{44}} & A_{K_{45}} \\ B_{K_5} \frac{C^2}{R_0 N} & B_{K_5} \frac{-C^2}{2N} & 0 & 0 & 0 & A_{K_{51}} & A_{K_{52}} & A_{K_{53}} & A_{K_{54}} & A_{K_{55}} \end{bmatrix}$$

(4.25)

where $A_{K_{ij}}$ is the (*i, j*) entry of the matrix $\boldsymbol{A_K}$ in eqn. (4.12) B_{K_i} is the i^{th} element of column vector $\boldsymbol{B_K}$ in eqn. (4.12); and C_{K_i} the i^{th} element of row vector $\boldsymbol{C_K}$ in eqn. (4.12).

4.3.3.2 Example

Our simulation study in Section 4.3.2 indicates that the acceptable controller can use $W_s(s) = \frac{s+11}{2s+11\times10^{-7}}$ and $W_T(s) = \frac{0.0031+3.5s}{1+0.01s}$ (i.e., with a_1= 0.00031, a_2= 3.5 and a_3=0.01). According to eqns. (4.7) and (4.23), the state-space representations of the weight functions are determined as follows.

$$W_s(s) = \left(\frac{A_{W_S}}{C_{W_S}}\middle|\frac{B_{W_S}}{D_{W_S}}\right) = \left(\frac{-w_b A}{w_b}\middle|\frac{1}{1/M}\right) = \left(\frac{-5.5\times10^{-7}}{5.5}\middle|\frac{1}{0.5}\right) \tag{4.26}$$

$$W_T(s) = \left(\frac{A_{W_T}}{C_{W_T}}\middle|\frac{B_{W_T}}{D_{W_T}}\right) = \left(\frac{-1/a_3}{a_2/a_3}\middle|\frac{1}{a_1/a_3}\right) = \left(\frac{-1/0.00031}{3.5/0.01}\middle|\frac{0}{0.031}\right) \tag{4.27}$$

In addition, the ST controller is obtained as

$$K(s) = \frac{1.522\times10^{-10}s^4 - 2.508\times10^{-7}s^3 - 2.201\times10^{-6}s^2 - 2.54\times10^{-7}s + 1.706\times10^{-6}}{2.23\times10^{-20}s^5 + 3.457\times10^{-10}s^4 + 0.05637s^3 + 0.5711s^2 + 0.819s - 2.855\times10^{-6}}$$

which can be rewritten in state-space as follows:

$$K(s) = \left(\begin{array}{ccccc|c} -1.95597413\times10^{3} & 0.86571 & -1.63354 & 8.34240\times10^{6} & 3.20567\times10^{10} & -9.152\times10^{-3} \\ -0.57281 & 0 & 0.00007 & 8.90634 & 3.3067\times10^{4} & 942.8549 \\ -3.07108\times10^{4} & -0.04755 & -1.99369 & -1.62608\times10^{3} & 6.0372\times10^{6} & 79.819986 \\ -0.90491\times10^{6} & -29.7482 & 24.7610 & -4.791770\times10^{5} & 1.7790361 & -2.63875\times10^{3} \\ -7.8852648\times10^{7} & -2.306725\times10^{2} & -4.50477\times10^{2} & -4.1754449\times10^{6} & 1.55024\times10^{10} & -2.59874\times10^{2} \\ \hline 1.59551 & -7.0747\times10^{-4} & 1.3350\times10^{-3} & -7.055925\times10^{3} & 2.6196\times10^{7} & 7.4803843\times10^{-6} \end{array}\right) \tag{4.28}$$

By substituting eqns. (2.26) to (4.28) in eqn. (4.25), one would obtain

$$A_{cl} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -38.74763 & -32.43055 & -12.3296 & 0 & 0 & 1.59551 & -7.1\times10^{-4} & 1.34\times10^{-3} & -7.055925\times10^{3} & 2.6196833\times10^{7} \\ -8.8409\times10^{6} & 1.1051\times10^{6} & 0 & -10^{-6} & 0 & 0 & 0 & 0 & 0 & 0 \\ 7.0526666\times10^{8} & -8.8158333\times10^{7} & 0 & 0 & -100 & 0 & 0 & 0 & 0 & 0 \\ 3.450291\times10^{4} & -4.31286\times10^{3} & 0 & 0 & 0 & -1.95597413\times10^{3} & 8.6571\times10^{-1} & -1.63354 & 8.634240\times10^{6} & 3.20567\times10^{10} \\ -3.554384\times10^{9} & 4.4429805\times10^{8} & 0 & 0 & 0 & -5.7281\times10^{-1} & 0 & 7\times10^{-5} & 8.90634 & 3.306702\times10^{4} \\ -3.009062\times10^{8} & 3.7613278\times10^{7} & 0 & 0 & 0 & -3.0710837\times10^{4} & -4.755\times10^{-2} & -1.99369 & -1.62608\times10^{3} & 6.037298\times10^{6} \\ 9.94758914\times10^{9} & -1.2434486\times10^{9} & 0 & 0 & 0 & -9.049148\times10^{6} & -29.74827 & 24.76105 & -4.791770\times10^{5} & 1.779036\times10^{9} \\ 9.79678476\times10^{8} & -1.22459809\times10^{8} & 0 & 0 & 0 & -7.8852648\times10^{7} & -2.30672\times10^{2} & -4.50477\times10^{2} & -4.175444\times10^{6} & -1.550240\times10^{10} \end{bmatrix}$$

The corresponding eigen-values are

$$\lambda = [-10^{-6} - 0.37004 \ \pm 0.98069\text{i} \ \ 1.02879 \pm 15.8058\text{i} \ \ -5.37234 \ \ -25.03 \ \ \ -100$$
$$-1.648098 \times 10^{8} \ \ -1.53380 \times 10^{10} \ ]$$

Since all the eigenvalues sit in the negative real part, $\boldsymbol{A_{cl}}$ is exponentially stable. According to Theorem 4.2, the proposed controller *K(s)* could be called *stabilizing*.

4.4 The STU Controller

As a general case of Sections 4.3 and 4.2, we shall now consider all three weight functions of $W_S(s)$, $W_T(s)$ and $W_U(s)$, along with *P(s*) to define the generalized plant function *G(s)*. The general configuration for STU controller has been given in Fig. 2.4. Like before, the plant for the TCP/AQM control system is $P(s) = \frac{\frac{R_0C^2}{2N^2}}{s+\frac{2N}{R_0^2C}} \times \frac{\frac{N}{R_0}}{s+\frac{1}{R_0}} e^{-sR_0}$ where *C* is the link capacity (packets/sec), *N* is the number of TCP sessions and R_0 is the RTT (Round Trip Time).

4.4.1 Controller Design

Like the SU controller and the ST controller, the weight functions $W_S(s)$, $W_U(s)$ and $W_T(s)$ need to be determined based on the information given in Section 3.2. The design procedure for STU controller would be similar to Fig. 3.1. Similar to Section 4.2.1.1 and Section 4.3.1.1, an example is provided on the selection of weight functions along with their effect on the performance of the TCP/AQM, thus demonstrating how our design choice is attained. To avoid repeating similar discussion, readers are referred to Appendix F for the design example for the STU controller.

4.4.1.1 Design Example

The same system parameters and configuration to Section 4.2.1.1 and Section 4.3.1.1 are used for the system in Fig. 4.1 where *N*=100 TCP sessions, R_0=0.25s, and *C*=9708 packets/s. Then we follow the iterative design procedure of Fig. 3.1 and use OPNET simulations to carry out Step 7. We also use *Robust Control*-toolbox in MATLAB by assigning the same weights of [α, β] = [1, 1] to $\|\boldsymbol{E}_\infty\|_\infty$ and$\|\boldsymbol{E_2}\|_2$, and choosing some relatively large numbers for δ_0 , ν_0 (e.g., $[\delta_0, \nu_0]$ = [200, 200]) in order for the iteration process converge to the optimum result. After some iterations (about 10), the weight functions $W_U(s)$, $W_T(s)$ and $W_s(s)$ are determined to be

$W_s(s) = \frac{s+11}{2s+11\times10^{-7}}$, $W_T(s) = \frac{0.001+6s}{1+0.01s}$ and $W_U(s) = 10^5$, and the STU controller is obtained as

$$K(s) = \frac{1.739\times10^{-10}s^4 + 3.563\times10^{-8}s^3 + 4.314\times10^{-7}s^2 + 1.243\times10^{-6}s + 3.699\times10^{-7}}{9.648\times10^{-5}s^5 + 0.0198\,s^4 + 0.253\,s^3 + 0.8189\,s^2 + 0.5147\,s + 2.648\times10^{-7}}$$

In the designed controller, the H_∞-norm of the closed-loop transfer functions from *w* to $\boldsymbol{Z}_\infty$ is determined to be $\|\boldsymbol{E}_\infty\|_\infty = 21.7344$ and H_2-norm of closed-loop transfer functions from $\boldsymbol{w}$ to $\boldsymbol{Z_2}$ is $\|\boldsymbol{E}\|_2 = 20.8166$. These norms give the minimum of the mixed H_2/H_∞ norm at $\gamma_{min} = 30.0951$. The bode diagram mentioned in Section 3.4 shows that our designed W_S, W_T and W_U are feasible functions as they meet the expected criteria mentioned in Section 3.2.

4.4.2 Effect of the Design Parameters

Similar to the SU and ST controllers, we shall investigate the impacts of the weight functions W_S, W_U and W_T on the system performance. The same procedure to the Sections 4.2.2 and 4.3.2 will be applied here to find the proper weight functions. To avoid repetition, readers may refer to Appendix F for details.

4.4.3 Stability Analysis

For the stability analysis of the proposed STU controller, the same method as in Sections 4.2.3 and 4.3.3 is used except the general plant model *G(s)* is different from the SU and the ST controller because it now contains all three weight functions of $W_S(s)$, $W_T(s)$ and $W_u(s)$ along with the plant *P(s).* The weight functions can be expressed in the form of "state-space function" as mentioned in eqns. (4.7), (4.8) and (4.23).

4.4.3.1 State-Space Model

Since the STU controller uses all three weight functions, the generalized plant model, $\boldsymbol{A_x}$, $\boldsymbol{B_2}$, $\boldsymbol{C_y}$ that are associated to the general plant model, and hence the closed-loop state matrix $\boldsymbol{A_{cl}}$ is different from eqns. (4.15, (4.16) and (4.24). Similar to procedure in Section 4.2.3.1, $\boldsymbol{A_x}$, $\boldsymbol{B_2}$ and $\boldsymbol{C_y}$ can be written as follows.

$$A_x = \begin{bmatrix} [A_P]_{3\times3} & [0]_{3\times2} \\ -C_P \begin{bmatrix} B_{W_s} \\ B_{W_T} \end{bmatrix} & \begin{bmatrix} A_{W_s} & 0 \\ 0 & A_{W_T} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ -\frac{4N}{R_0^4 C} & -\left(\frac{6N}{R_0^3 C} + \frac{2}{R_0^2}\right) & -(\frac{2N}{R_0^2 C} + \frac{3}{R_0}) & 0 & 0 \\ -\frac{C^2}{R_0 N} & \frac{C^2}{2N} & 0 & -w_b A & 0 \\ -\frac{C^2}{R_0 N} & \frac{C^2}{2N} & 0 & 0 & -\frac{1}{a_3} \end{bmatrix}$$

$$B_2 = \begin{bmatrix} [B_P]_{3\times1} \\ -B_{W_S} D_P \\ B_{W_T} D_P \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$C_y = [[-C_P]_{1\times3} \quad 0 \quad 0] = [\frac{C^2}{R_0 N} \quad \frac{-C^2}{2N} \quad 0 \quad 0 \quad 0] \qquad (4.29)$$

Therefore, by using (4.14) $\boldsymbol{A_{cl}}$ can be presented by:

$$A_{cl} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{4N}{R_0^4 C} + D_k \frac{C^2}{R_0 N} & -\left(\frac{6N}{R_0^3 C} + \frac{2}{R_0^2}\right) - D_k \frac{C^2}{2N} & -(\frac{2N}{R_0^2 C} + \frac{3}{R_0}) & 0 & 0 & C_{K_1} & C_{K_2} & C_{K_3} & C_{K_4} & C_{K_5} \\ -\frac{C^2}{R_0 N} & \frac{C^2}{2N} & 0 & -w_b A & 0 & 0 & 0 & 0 & 0 & 0 \\ -\frac{C^2}{R_0 N} & \frac{C^2}{2N} & 0 & 0 & -\frac{1}{a_3} & 0 & 0 & 0 & 0 & 0 \\ B_{K_1} \frac{C^2}{R_0 N} & B_{K_1} \frac{-C^2}{2N} & 0 & 0 & 0 & A_{K_{11}} & A_{K_{12}} & A_{K_{13}} & A_{K_{14}} & A_{K_{15}} \\ B_{K_2} \frac{C^2}{R_0 N} & B_{K_2} \frac{-C^2}{2N} & 0 & 0 & 0 & A_{K_{21}} & A_{K_{22}} & A_{K_{23}} & A_{K_{24}} & A_{K_{25}} \\ B_3 \frac{C^2}{R_0 N} & B_{K_3} \frac{-C^2}{2N} & 0 & 0 & 0 & A_{K_{31}} & A_{K_{32}} & A_{K_{33}} & A_{K_{34}} & A_{K_{35}} \\ B_{K_4} \frac{C^2}{R_0 N} & B_{K_4} \frac{-C^2}{2N} & 0 & 0 & 0 & A_{K_{41}} & A_{K_{42}} & A_{K_{43}} & A_{K_{44}} & A_{K_{45}} \\ B_{K_5} \frac{C^2}{R_0 N} & B_{K_5} \frac{-C^2}{2N} & 0 & 0 & 0 & A_{K_{51}} & A_{K_{52}} & A_{K_{53}} & A_{K_{54}} & A_{K_{55}} \end{bmatrix}$$

(4.30)

where $A_{K_{ij}}$ is the (*i,j*) entry of the matrix $\boldsymbol{A_K}$ in eqn. (4.12) B_{K_i} is the i^{th} element of column vector $\boldsymbol{B_K}$ in eqn. (4.12); and C_{K_i} the i^{th} element of row vector $\boldsymbol{C_K}$ in eqn. (4.12).

4.4.3.2 Example

As mentioned in Section 4.4.1.1, the STU controller has been designed for the TCP/AQM

network (N=100, R_0=0.25s, C=9708 packet/second) while the weight functions are determined as $W_s(s) = \frac{s+11}{2s+11\times10^{-7}}, W_T(s) = \frac{0.001+6s}{1+0.01s}$ and $W_U(s)$=10^5. According to eqns. (4.7), (4.8) and (4.23), state-space representation of the weight functions can be expressed as follows.

$$W_s(s) = \left(\frac{A_{W_S}}{C_{W_S}}\middle|\frac{B_{W_S}}{D_{W_S}}\right) = \left(\frac{-w_b A}{w_b}\middle|\frac{1}{1/M}\right) = \left(\frac{-5.5\times10^{-7}}{5.5}\middle|\frac{1}{0.5}\right) \quad (4.31)$$

$$W_T(s) = \left(\frac{A_{W_T}}{C_{W_T}}\middle|\frac{B_{W_T}}{D_{W_T}}\right) = \left(\frac{-1/a_3}{a_2/a_3}\middle|\frac{1}{a_1/a_3}\right) = \left(\frac{-100}{600}\middle|\frac{0}{0.1}\right) \quad (4.32)$$

$$W_U(s) = \left(\frac{A_{W_U}}{C_{W_U}}\middle|\frac{B_{W_U}}{D_{W_U}}\right) = \left(\frac{0}{0}\middle|\frac{0}{W_U}\right) = \left(\frac{0}{0}\middle|\frac{0}{10^5}\right) \quad (4.33)$$

For the designed STU controller, we obtain its transfer function in "state-space form" as follows

$$K(s) = \left(\begin{array}{ccccc|c} -7.8314\times10^{-5} & 1.9903\times10^{-3} & -4.8983\times10^{-5} & -1.04721\times10^{-4} & -3.36703\times10^{-3} & -179.48487 \\ 1.5623\times10^{-1} & -4.4492 & 1.65409\times10^{-1} & 3.46008\times10^{-1} & 13.83428 & -37.3889 \\ -2.669282 & 75.6288 & -5.7950 & -8.3109 & -187.01847 & 1585.212 \\ 7.39795 & -141.0277 & -8.3873 & -9.418776 & -350.1602 & -1337.1687 \\ 3.41303\times10^{-1} & -6.4159 & -1.477 & -2.646311 & -185.4066 & 16.13710 \\ \hline -2.32853\times10^{-9} & -2.092092\times10^{-8} & 3.35846\times10^{-10} & -8.00169\times10^{-11} & -2.27890\times10^{-9} & 0 \end{array}\right) \quad (4.34)$$

Substituting eqns. (4.31) to (4.34) in eqn. (4.30), one would obtain

$$A_{cl} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -10.548 & -35.9555 & -12.3296 & 0 & 0 & 2.32853\times10^{-9} & -2.092092\times10^{-8} & 3.35846\times10^{-10} & -8.00169\times10^{-11} & -2.27890\times10^{-9} \\ -8.8409\times10^{6} & 1.1051\times10^{6} & 0 & -10^{-6} & 0 & 0 & 0 & 0 & 0 & 0 \\ 9.2341\times10^{8} & -1.15426\times10^{8} & 0 & 0 & -100 & 0 & 0 & 0 & 0 & 0 \\ 6.7662\times10^{8} & -8.457\times10^{7} & 0 & 0 & 0 & -7.8314\times10^{-5} & 1.9903\times10^{-3} & -4.8983\times10^{-5} & -1.04721\times10^{-4} & -3.36703\times10^{-3} \\ 1.4094\times10^{8} & -1.76186\times10^{7} & 0 & 0 & 0 & 1.5623\times10^{-1} & -4.4492 & 1.65409\times10^{-1} & 3.46008\times10^{-1} & 13.83428 \\ -5.9759\times10^{9} & 7.4699\times10^{8} & 0 & 0 & 0 & -2.669282 & 75.6288 & -5.7950 & -8.3109 & -187.01847 \\ 5.040872\times10^{9} & -6.30109\times10^{8} & 0 & 0 & 0 & 7.39795 & -141.0277 & -8.3873 & -9.418776 & -350.1602 \\ -6.083\times10^{7} & 7.60422\times10^{6} & 0 & 0 & 0 & 3.41303\times10^{-1} & -6.4159 & -1.477 & -2.646311 & -185.4066 \end{bmatrix}$$

For the $\boldsymbol{A_{cl}}$ introduced above, its eigen values are

$$\lambda = [-5.5\times10^{-7} \quad -100 \quad -191.626 \quad -8.7657 \quad -7.9609 \quad -0.33940+0.28i \quad -0.33940-0.28i \quad -0.34728 \quad -4.1566 \quad -3.8639]$$

All these eigenvalues have negative real parts. Therefore, $\boldsymbol{A_{cl}}$ is exponentially stable, and the controller $K(s)$ is *stabilizing* according to Theorem 4.2.

4.5 Performance Evaluation

In this section, we shall use OPNET simulation [Opnt13] to verify the capability of our designed controllers, and the extent of improvement over some existing controller.

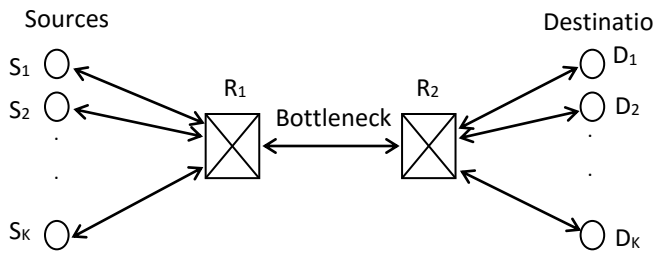


Fig. 4.12: Network Simulation Topology

4.5.1 Simulation Model and Set up

We verify the designed controller in the network topology shown in Fig. 4.12. Each S_i is a TCP-Reno source connected to router R_1 and each D_i is a destination downstream from router R_2. Router 1 becomes the bottleneck in the network when the total sending rate of the sources exceeds the bandwidth of the bottleneck link between R_1 and R_2. On the other hand, Router 2 is configured to have a sufficiently high service rate to process the arrived traffic so that congestion will never happen there. We are interested in investigating the controller behavior in the most congested router.

We use the same network parameters mentioned in previous sections. Since the OPNET modeler [Opne13] is packet-based, we need to convert the s-domain controller into z-domain in order to run in a discrete-time environment. Our experiments are conducted using OPNET Modeler 17.5 [Opne13] on an Intel Core i5 platform with a 2.30GHz CPU. The simulation time would depend on the bottleneck bandwidth and the simulated time. For a typical simulated time of 200 seconds in our experiments, the simulation time usually takes about 3 minutes during which an order of 1 million packets is collected for our statistics.

The controllers are evaluated by the following performance measures:

1) Queue size is the length of the bottleneck buffer queue (measured in packets) seen by a departing packets. The mean queue length and queue length standard deviation are used to compare different queue length. Since we are showing the time evolution, it is not required to calculate the queue length interval confidence.

2) Queueing delay is the waiting time of a packet in the router queue before its service. Measurements are taken from the time the first bit of a packet is received at the queue until the time the first bit of the packet is transmitted over the channel.
3) Packet loss rate is defined as the ratio between the number of packet dropped and the number of total packets received every second at the bottleneck.
4) Source throughput (or source sending rate) is defined to be the number of bits (or packets) successfully sent out by a source per second, i.e. bits/second (or packets/second) [Opne13]. Later we will use throughput and sending rate interchangeably.
5) Goodput is defined as the rate of sending useful information in bps by each source. The amount of data excludes protocol overhead as well as retransmitted data packets. It is a measure of useful data at destinations.
6) Rise time is the time required for the measured value (e.g. queue length) to go from 10% to 90% of the targeted (controlled value) value *F* (e.g. the controlled queue length of 200 often used in this research). The fall time is the time the measured value takes to drop from $F + 0.9 \times (P - F)$ to $F + 0.1 \times (P - F)$ where *P* is the peak value. Since the measured value may still have fluctuations at steady state, we shall take the first crossing of the 90% (10%) of the targeted value to measure the rise (fall) time.
7) Peak Value is the maximum value of a variable during its transient state before settling down to its steady-state.

Note that the parameters *N*, R_o and *C* are designed parameters. They all take on a nominal value for a given designed controller. Corresponding to them are the instantaneous values under a changing network scenario. We shall use the same *N* to refer to the instantaneous values later if there is no confusion. For example, Varying *N* is understood to be "Varying Network Load" when the number of connections is changing even if the controller is designed for a nominal *N* value (say *N*=100). Likewise, the N values in the figures in Section 4.5.2.9 are for the instantaneous number.

4.5.2 Performance Comparison

We would like to investigate the instantaneous queue size, queueing delay, throughput, goodput and packet loss ratio of our designed controllers. Our buffer size is 1000 packets. Note that the reference queue size cannot be zero as this would cause a sever restriction in sending rate, nor it can be set at 1000 packets as any fluctuations would cause packet loss. We choose 200 packets as a compromise after some experiments. We also compare them with the SU H_∞ controller, the STU H_∞ controller and the PI controller to show the performance improvements achieved by our controller. All three controllers are designed for a TCP/AQM system using the same plant model *P(s)* as ours. The PI controller [HoMi02] controls the packet drop probability δ_p (See Fig. 4.1) with a controller $K(s) = k_P + \frac{k_I}{s} = 2.8 \times 10^{-6} + \frac{2.3\times10^{-7}}{s}$. The H_∞ theory is used to design both the SU H_∞ and the STU H_∞ controllers. The SU H_∞ controllers [ChYa05] uses a Sensitivity weight functions $W_s = \frac{s/1.5+3.5}{s+3.5\times10^{-4}}$ and an Input Sensitivity weight function $W_u = 5 \times 10^5$. The STU H_∞ [ChYa07] uses $W_s = \frac{s/1.5+3.5}{s+3.5\times10^{-4}}$, $W_u = 5 \times 10^5$ and a complementary sensitivity weight function $W_T(s) = \frac{2.4+0.56s}{1+10^{-3}s}$. The network parameters of these three controllers are the same as our network which are *C*=9708 packets/second, *N*=100, R_0=0.25 ms and an average packet size of 576 bytes. In addition, we compare their performance under varying scenarios of sudden bandwidth changes, varying RTT delay and varying *N* as well as having uncontrolled traffic towards the end of this chapter.

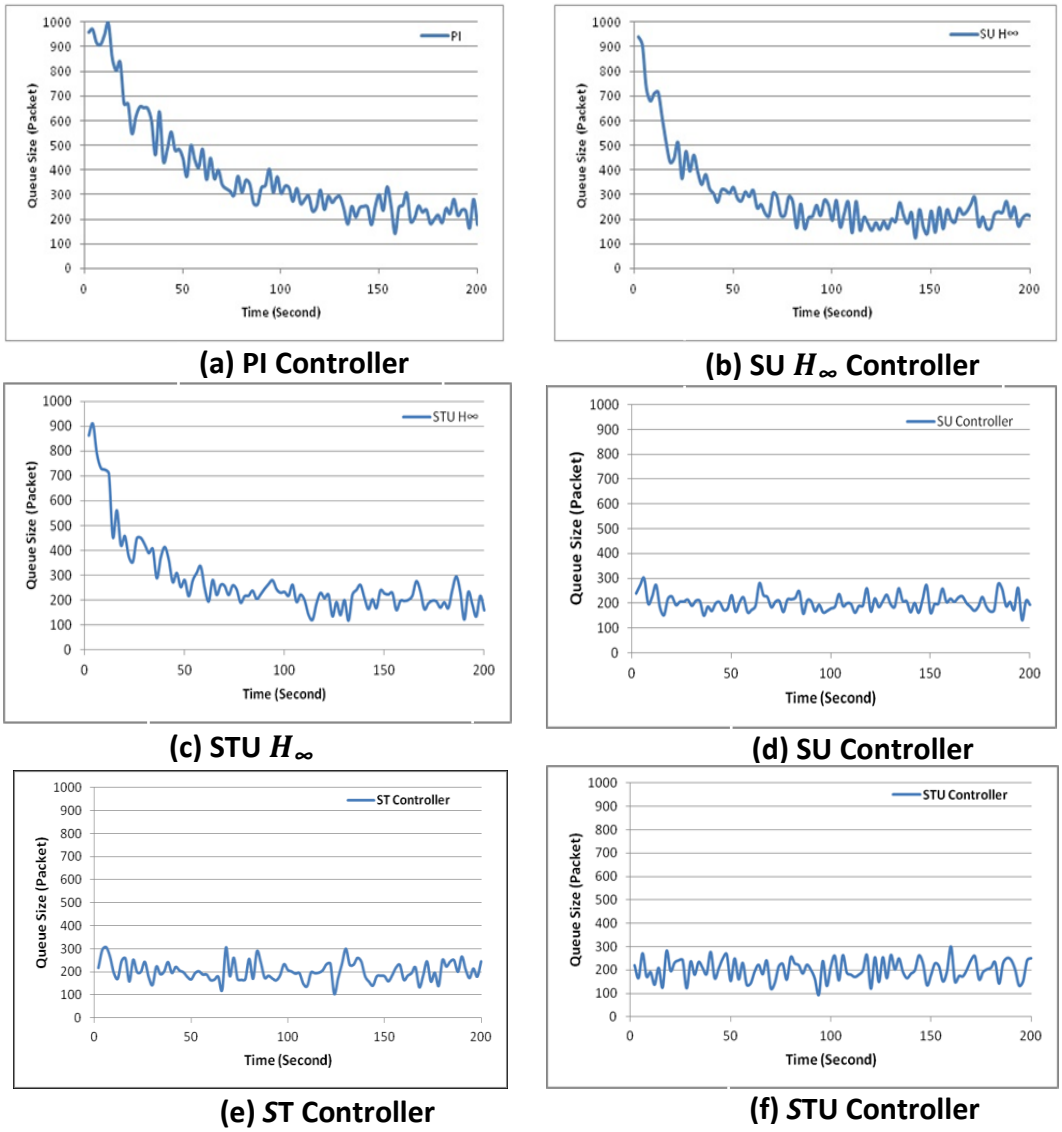
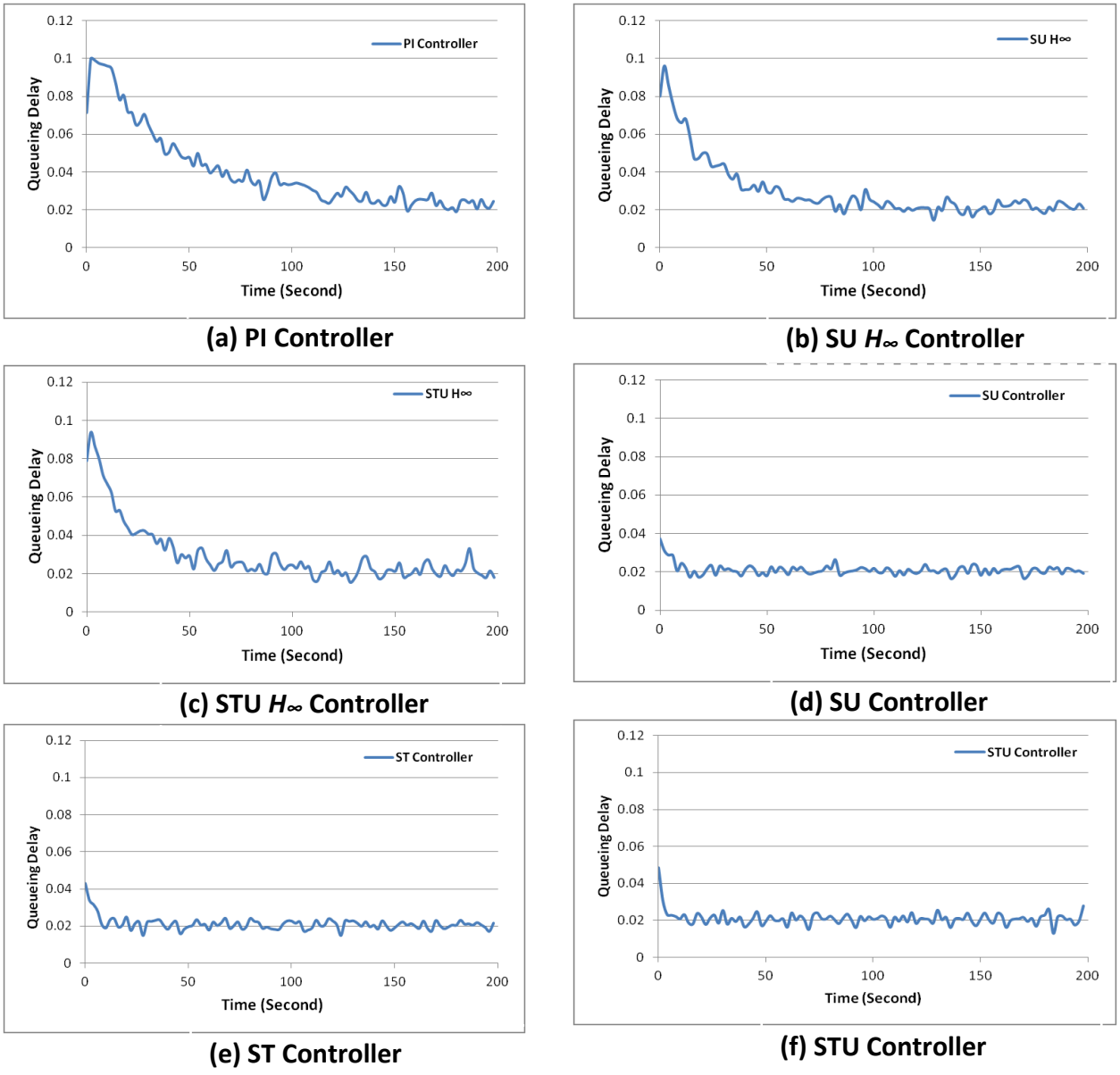


Fig. 4.13: Instantaneous Queue Size Comparison of Different Controllers

4.5.2.1 Queue Size

Fig. 4.13 compares instantaneous queue size among three different controllers. It is shown that the PI controller needs 132 seconds to "settle" to (first reaching) the desired queue size level of 200 packets, and the queue size exceeds the maximum buffer size of 1000 packets in the beginning when some packets are lost. The SU H_∞ and STU H_∞ controllers have a shorter rise time (~ 80 sec and 60sec respectively) and a smaller queue size peak value (932 packets and 910 respectively) than the PI controller. In comparison, our proposed controllers provide significant improvement in both the transient response and steady state level of the queue size.

The proposed SU controller has the smaller queue size peak values (~290packets) and the shorter rise/fall times (~6s). The proposed ST controller has larger peak queue size than the SU controller (~306 packets). Among all controllers, the STU controller has the smallest queue size peak value (~245 packets) and the shortest rise/fall time (~2s). Furthermore, the queue size oscillation of our designed controllers are considerably smaller than those of the STU H_∞, the STU H_∞ and the PI controller.



(a) PI Controller

(b) SU H_∞ Controller

(c) STU H_∞ Controller

(d) SU Controller

(e) ST Controller

(f) STU Controller

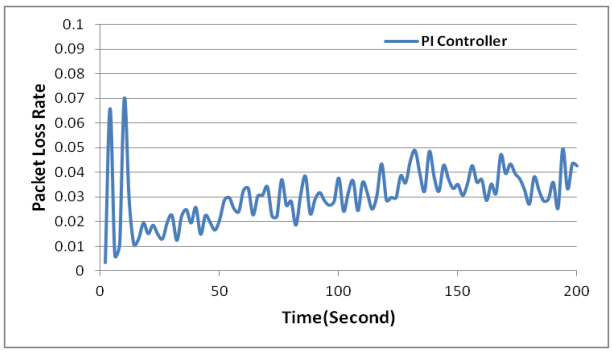
Fig. 4.14: Queueing Delay Performance Comparison

4.5.2.2 Queueing Delay

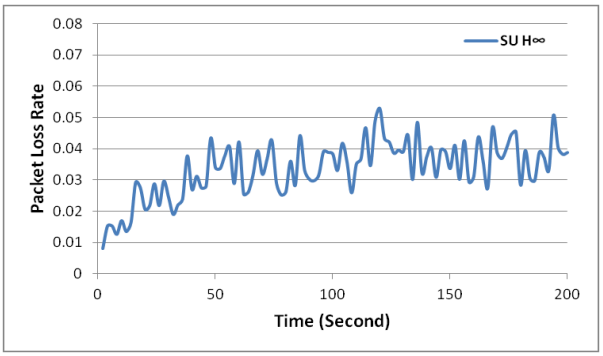
Fig. 4.14 shows the queueing delay for the designed controllers. All the controllers attain the same queueing delay in the steady state. However, it is apparent from this figure that there are significant differences in the beginning among our proposed H_2/H_∞ controllers (the SU, ST and STU controllers) and other controllers (the SU H_∞, STU H_∞ and the PI controllers). The PI controller has the worst performance with the longest queueing delay (~0.1 sec) and rise time (~180 sec). The SU H_∞ and STU H_∞ controllers show slightly better performance than PI controllers with a smaller value for the queueing delay in the beginning (~ 0.095s and 0.093s respectively) and rise/fall time (~80s and 78s respectively). Our H_2/H_∞ controllers have the significantly better performance with the queueing delay in the beginning (~ 0.04s, 0.04s and 0.05s for the SU, the ST and the STU controller respectively) and rise/fall time (~ 8s, 8s and 9s for the SU, the ST and the STU controller respectively). One can see that all of our designed H_2/H_∞ controllers have almost similar queueing delay performance.

4.5.2.3 Packet Loss Rate

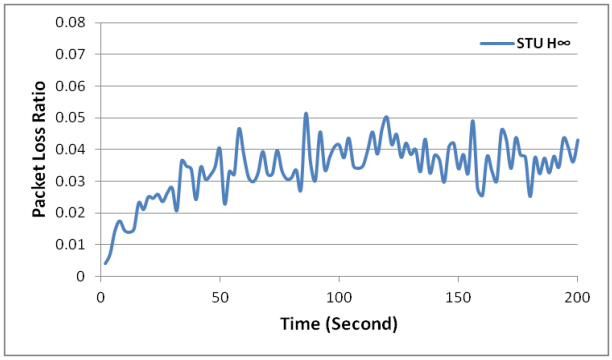
Fig. 4.15 below presents packet loss rates of different controllers. The PI controller has large oscillations (between 0.003 and 0.07) while the packet loss rate increases slowly as time evolves. It reaches 0.035 after 130 seconds, which is the same as the queue size rise/fall time shown in Fig. 4.12. The SU H_∞ controller has a smaller packet loss rate (around 0.01) in the beginning (which has resulted in the big queue size observed in Fig. 4.13). The loss rate reaches a steady state of around 0.03 after 45s and. The STU H_∞ controller has smaller packet loss rate at the beginning (~0.005) and reaches to a steady state of around 0.03 after 40s. The SU, ST and STU controller have a packet loss rate around 0.035 but exhibits a faster rise/fall time (~2s, 3s, 9s respectively) with oscillations smaller than other controllers around the steady value. Note that we show the y-axis in a very large scale of up to 0.08 only since it is very rare to have loss values higher than 0.08.



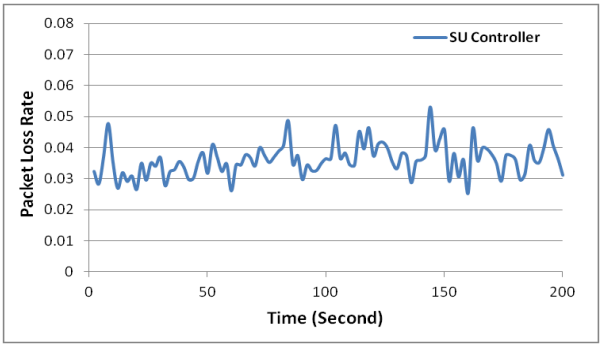
(a) PI Controller



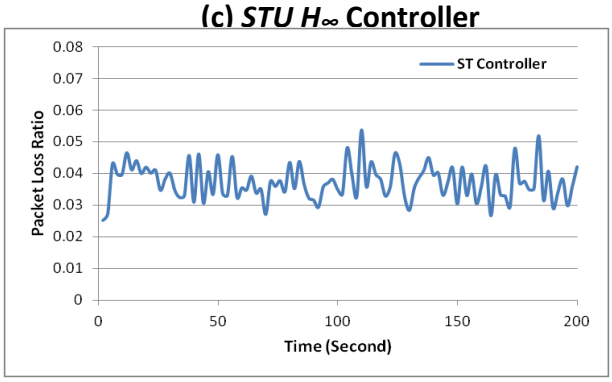
(b) *SU H∞* Controller



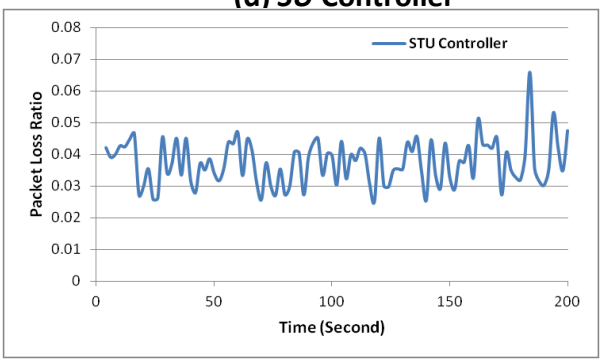
(c) *STU H∞* Controller



(d) *SU* Controller

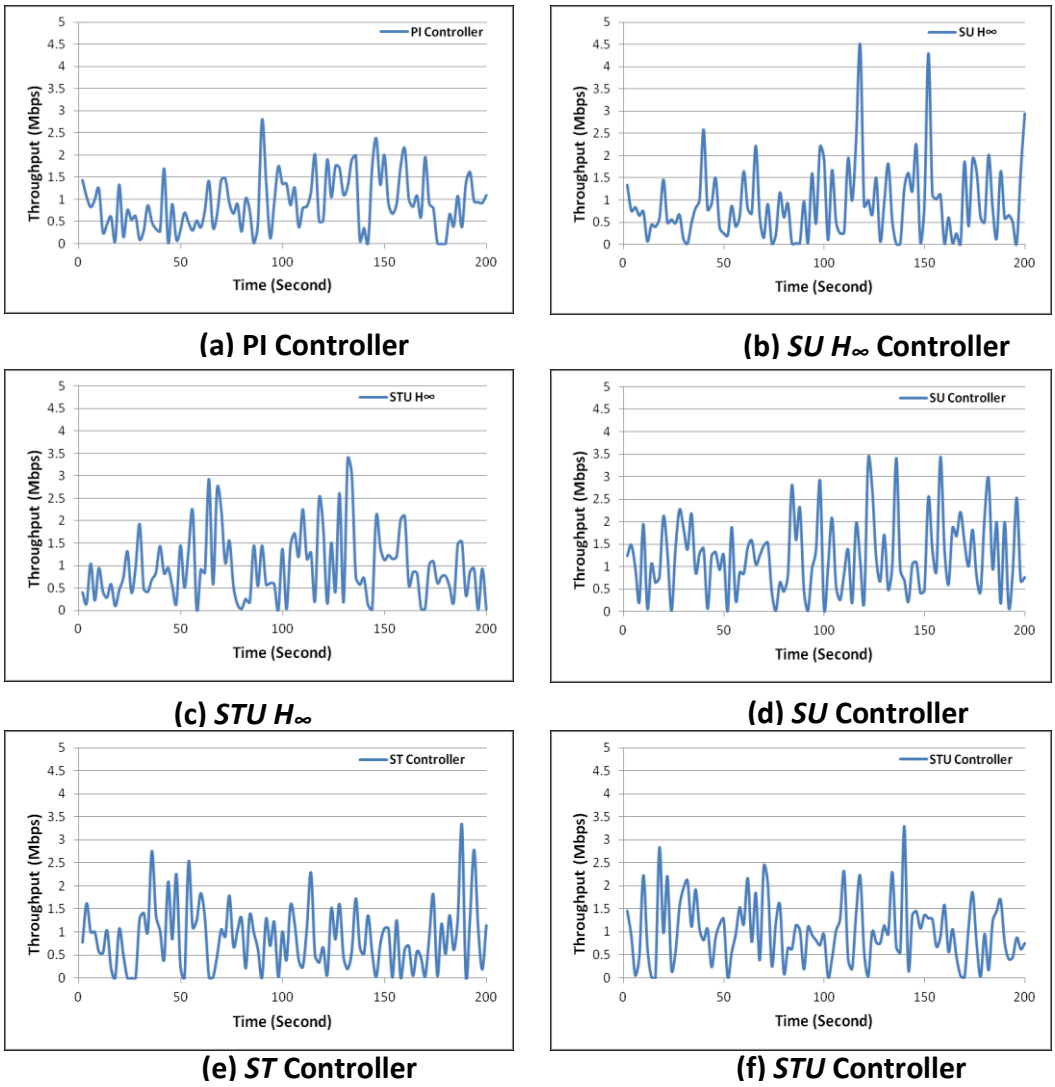


(e) *ST* Controller



(f) *STU* Controller

Fig. 4.15: Packet Loss Rate Comparison of Different Controllers



(a) PI Controller **(b) *SU* H_∞ Controller**

(c) *STU* H_∞ **(d) *SU* Controller**

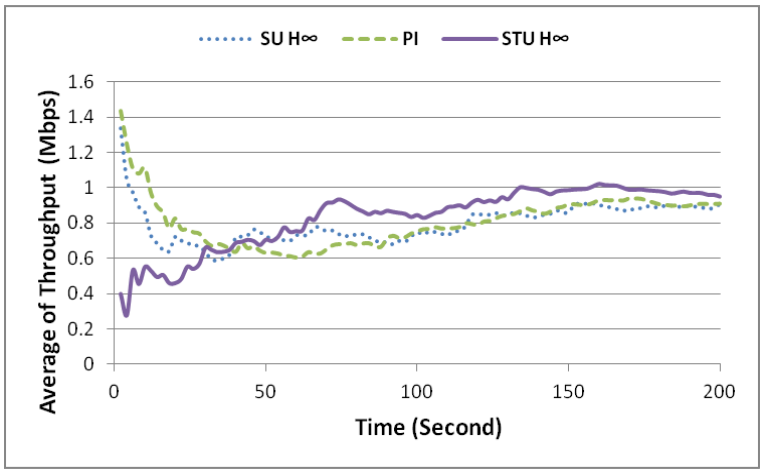
(e) *ST* Controller **(f) *STU* Controller**

Fig. 4.16: Comparison of Throughput Performance for Different Controller

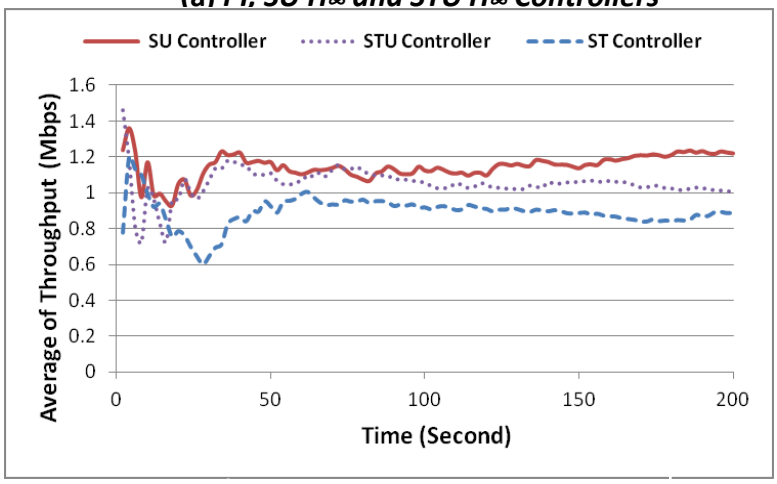
4.5.2.4 Throughput

Fig. 4.16 shows an example of the throughput (source sending rate) of one of the ftp sources (e.g. ftp 11) for the proposed H_2/H_∞ controllers as well as the SU H_∞, STU H_∞ and the PI controllers. Since all controllers are window-based controllers, sources sending rates are not smooth. In the window-based controllers, the congestion window size and hence the sending rate are changed upon a congested packet or dropped one. However, we can see that the proposed H_2/H_∞ controllers have slightly higher throughput than other controllers. In the interest of showing more clearly, Fig. 4.17 shows the average value of the throughput for all controllers. One can see that SU controllers has almost 37% more throughput in average than PI

and SU H_∞. In addition, although the STU H_∞ controller has similar throughout to the STU controller at the end, it has less average throughput than other controllers at the beginning.



(a) *PI, SU H_∞ and STU H_∞ Controllers*



(b) *SU, ST and STU* Controllers

Fig. 4.17: Average Throughput Performance Comparison

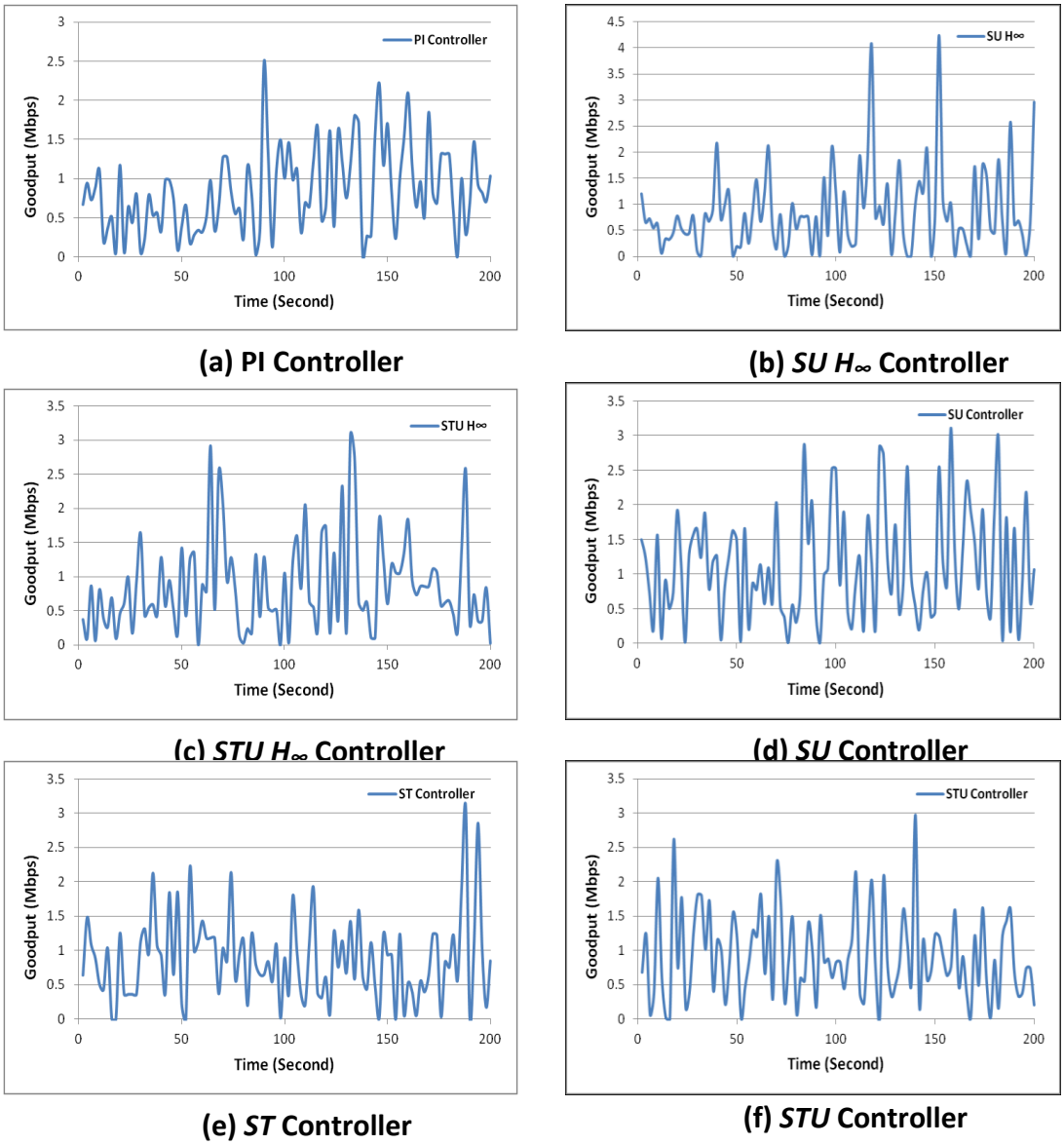
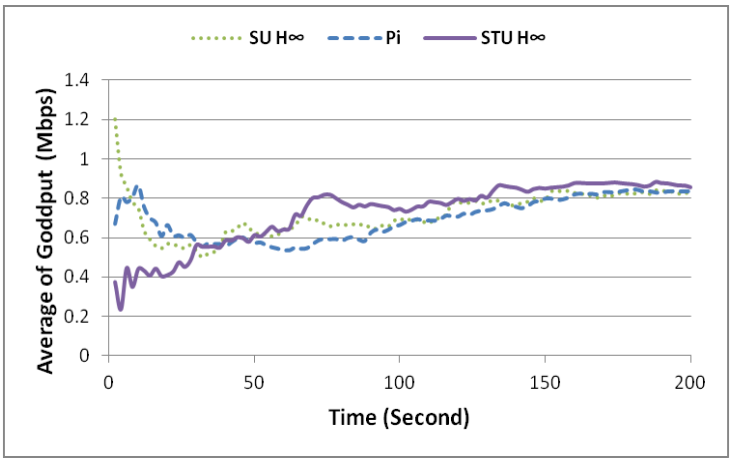


Fig. 4.18: Goodput Performance Comparison

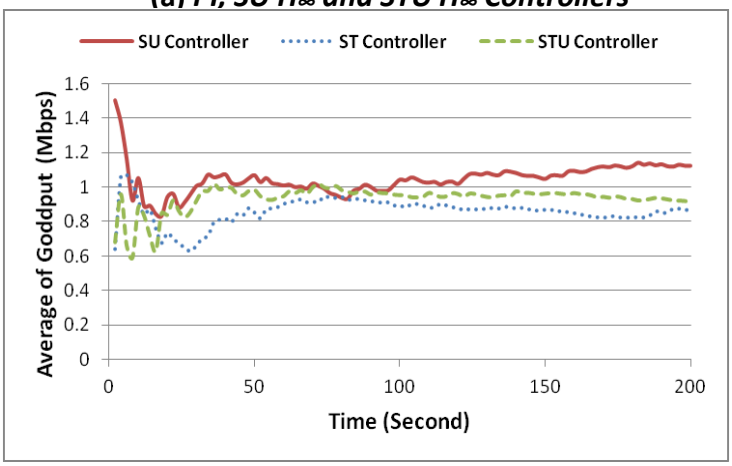
4.5.2.5 Goodput

Fig. 4.18 shows the good-put comparison of the SU, STU, ST, SU H_∞, STU H_∞ and the PI controller for an example source (ftp 11). Similar to the throughput, the proposed H_2/H_∞ controllers propose higher data goodput rate than the PI, the SU H_∞ and STU H_∞ controllers. In order to show it more clearly, Fig. 4.19 shows the average of goodput values for different controllers. It shows that for the same source, the system with proposed controllers give higher goodput (with a mean at 1.12 Mbps for the SU controller, 0.95Mbps for the STU controller and

0.9Mbps for the ST controller) than the SU H_∞ (mean value at 0.84 Mbps), the STU H_∞ (mean value 0.83Mbps) and PI controller (0.83 Mbps).



(a) *PI, SU H_∞ and STU H_∞ Controllers*



(b) *SU, ST and STU*

Fig. 4.19: Average Goodput Performance Comparison

4.5.2.6 Sudden Bandwidth Changes

Since a deterministic amount of bandwidth is usually not available in connection-based network such as the wireless network, the bottleneck link can change suddenly. Therefore, we would like to investigate the robustness capability of the designed H_2/H_∞ controllers to sudden bandwidth changes.

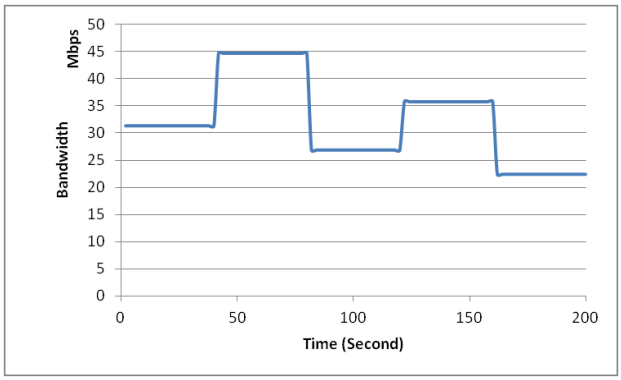
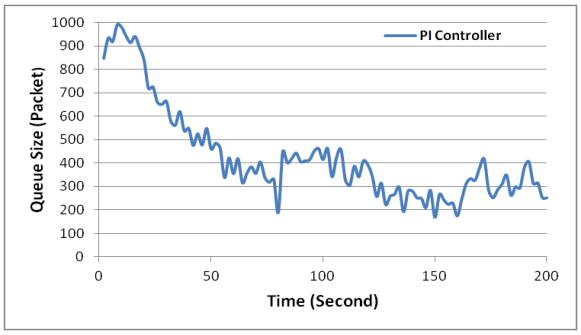
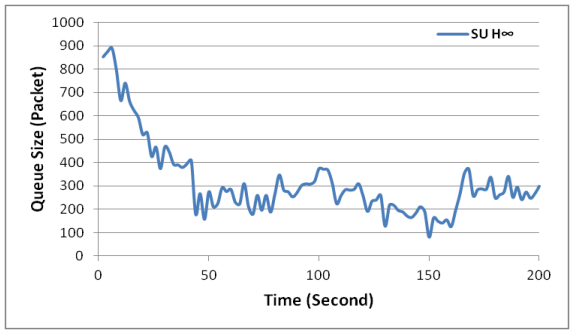


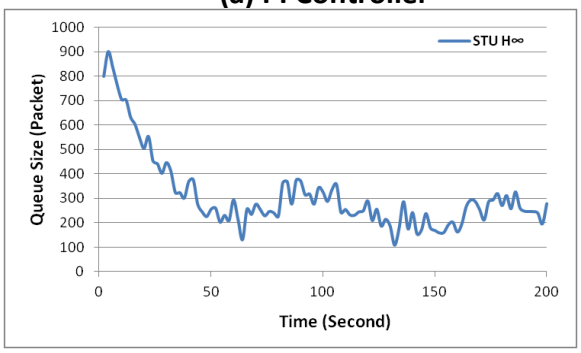
Fig. 4.20: Bandwidth Variation



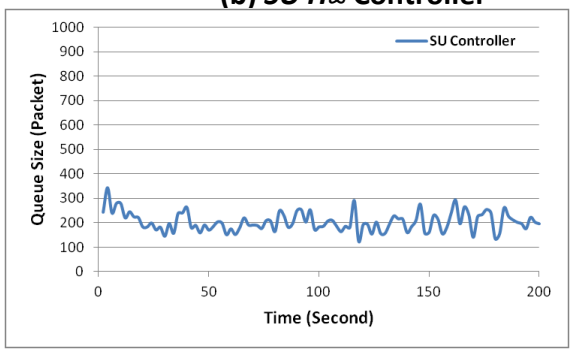
(a) PI Controller



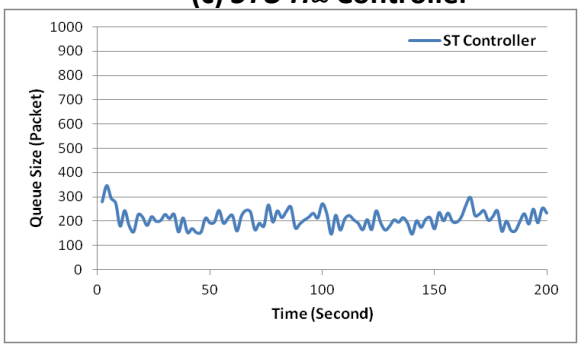
(b) *SU* H_∞ Controller



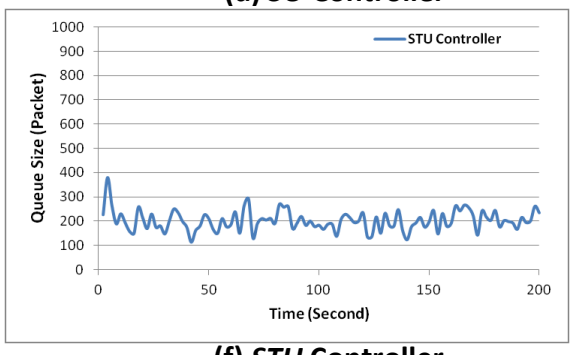
(c) *STU* H_∞ Controller



(d) *SU* Controller



(e) *ST* Controller



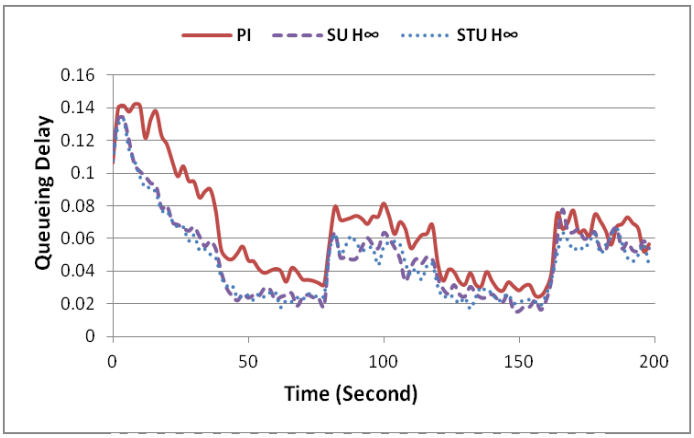
(f) *STU* Controller

Fig. 4.21: Instantaneous Queue Size Comparison of Different Controllers under Bandwidth Changes

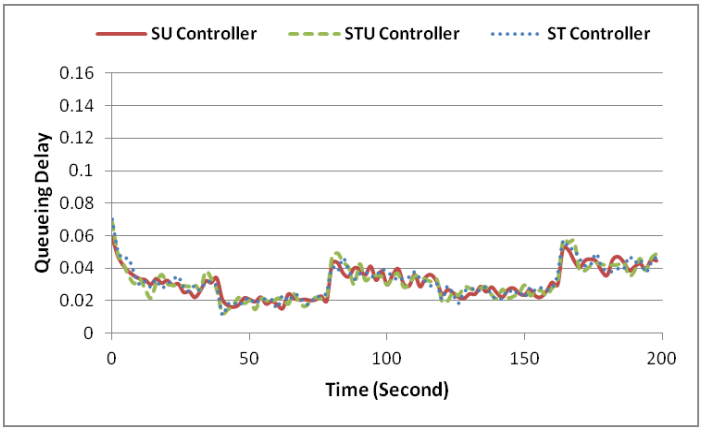
Unlike the constant bandwidth of T3 rate (44,736,000bps), the bottleneck bandwidth is allowed to make step changes. Fig. 4.20 is one such scenario where the bandwidth varies in steps between 22 and 45 Mbps. The biggest change occurs at t=80s when the bandwidth drops from 45 Mbps to 27 Mbps (a 40% change). We shall use this scenario to study and compare the robustness of the controllers under study.

Fig. 4.21 compares the queue size of different controllers under bandwidth changes. It can be seen that the queue size in router 1 cannot reach to desired/designed value of 200 packets for the PI, SU H_∞ and STU H_∞ controllers. This is because they have long rise/fall time that queue value cannot reach to steady state value before the next bandwidth change is applied to the system. On the other hand, the SU, ST and STU controllers are able to maintain the queue size to the desired level of ~200 packets. Note that the queue sizes for the SU, ST and STU controller hardly changes at the time instants of bandwidth changes. This can be attributed to the short rise/fall time of these controllers. Therefore, using H_2/H_∞ controllers for the system leads to robust system under bottleneck bandwidth variation.

Fig. 4.22 compares queueing delay of different controllers under the bandwidth changes. Since the H_2/H_∞ controller has the smallest rise/fall time, the controllers can reach to state values before the new bandwidth value is applied to the link. Also, they have smaller step fluctuation suggesting that the H_2/H_∞ controllers are more robust to bottleneck bandwidth changes. One can also see that there is only a slight difference among the H_2/H_∞ controllers reaction to step bandwidth changes.



(a) *PI, SU H∞ and STU H∞*

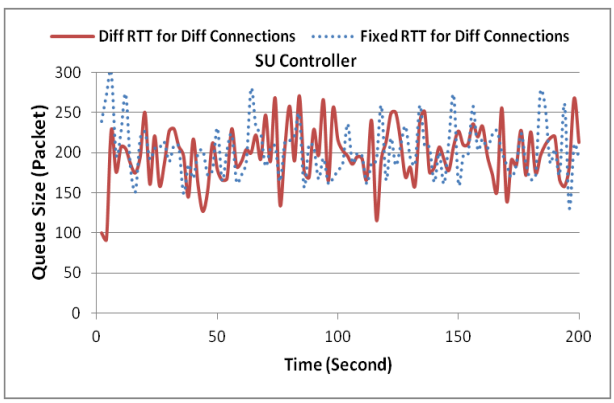


(b) *SU, ST and STU*

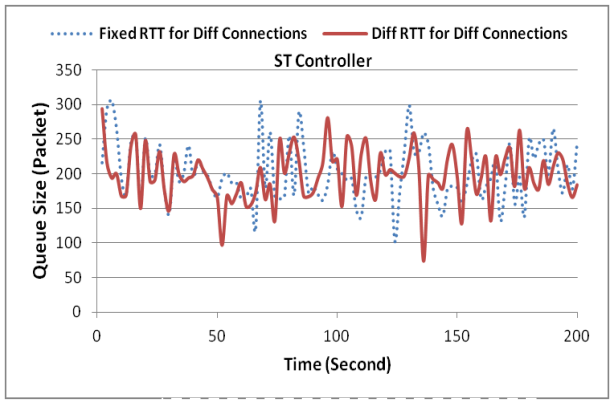
Fig. 4.22: Queueing Delay for the SU, ST and STU Controllers under Bandwidth Changes

4.5.2.7 Different RTTs for Different Connections

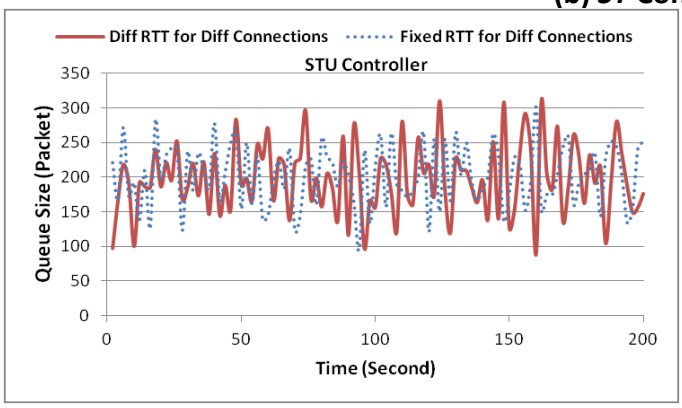
Since connections may not have the same RTT, we would also like to investigate the robustness of our designed controllers when different connections may have different RTTs. We do this by assigning different RTPD (Round Trip Propagation Delay) for different connections. The RTPD includes the forward path propagation delay and the feedback propagation delay, but does not include the queueing delay. There are 100 ftp connections and 20 http connections in this investigation. For the ftp sources, the RTDPs are 130 ms, 100 ms, 160 ms, 190 ms and 220 ms for the connections 1~20, 21~40, 41~60, 61~80 and 81~100 respectively. For the http sources the RTDPs are 150 ms and 200 ms for connections 1~10 and 11~20 respectively.



(a) *SU* Controller



(b) *ST* Controller



(c) *STU* Controller

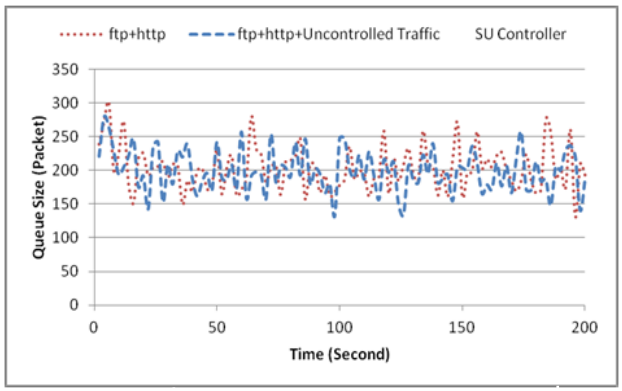
Fig. 4.23: Queue Size of the SU, the ST and the STU Controllers Under Different RTT for Different Connections

Fig. 4.23 compares the queue size for the designed controllers under the scenario where the RTTs are different for different connections. It can be seen from the figure that all the designed H_2/H_∞ controllers has reasonable peak queue values (260 packets for the SU controller, 290 packets for the ST controller and 305 packets for the STU controller). They are able to reach the desired queue size in a small rise/fall time (~5 sec for the SU controller, 4sec for the ST controller and 6sec for the STU controller) with relatively the same oscillation when compared to the fixed RTT scenario (blue dotted line) .

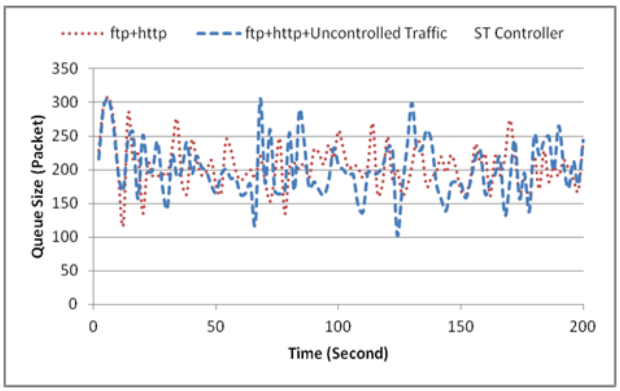
4.5.2.8 Disturbance from Uncontrolled Traffic

In order to investigate the behavior of our designed controllers under uncontrolled traffic, we introduce another 20 uncontrolled ftp (unresponsive UDP-like) flows into a network already with 80 long-lived ftp and 20 short-lived http flows. Each uncontrolled ftp source has a

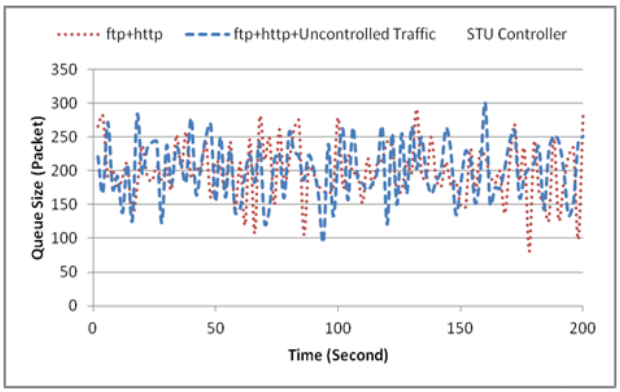
relatively large fixed Reno congestion widow *cwnd*=100 (Congestion Window Size) so that the transmitted traffic is not influenced/controlled by any congestion information.



(a) *S*U Controller



(b) *S*T Controller



(c) *S*TU Controller

Fig 4.24: Queue Size of the SU, ST and STU Controllers under Uncontrolled Traffic Sources

Fig. 4.24 shows the queue size of the designed controllers for different source's flows. The designed H_2/H_∞ controllers can still work reasonably in the presence of uncontrolled traffic in the network; they are able to reach to the desire queue size in a very short transient time. Therefore, the system can guarantee assigning a portion of bandwidth to the uncontrolled traffic without having effect on the system performance.

4.5.2.9 Network Load Other Than The Nominal Value

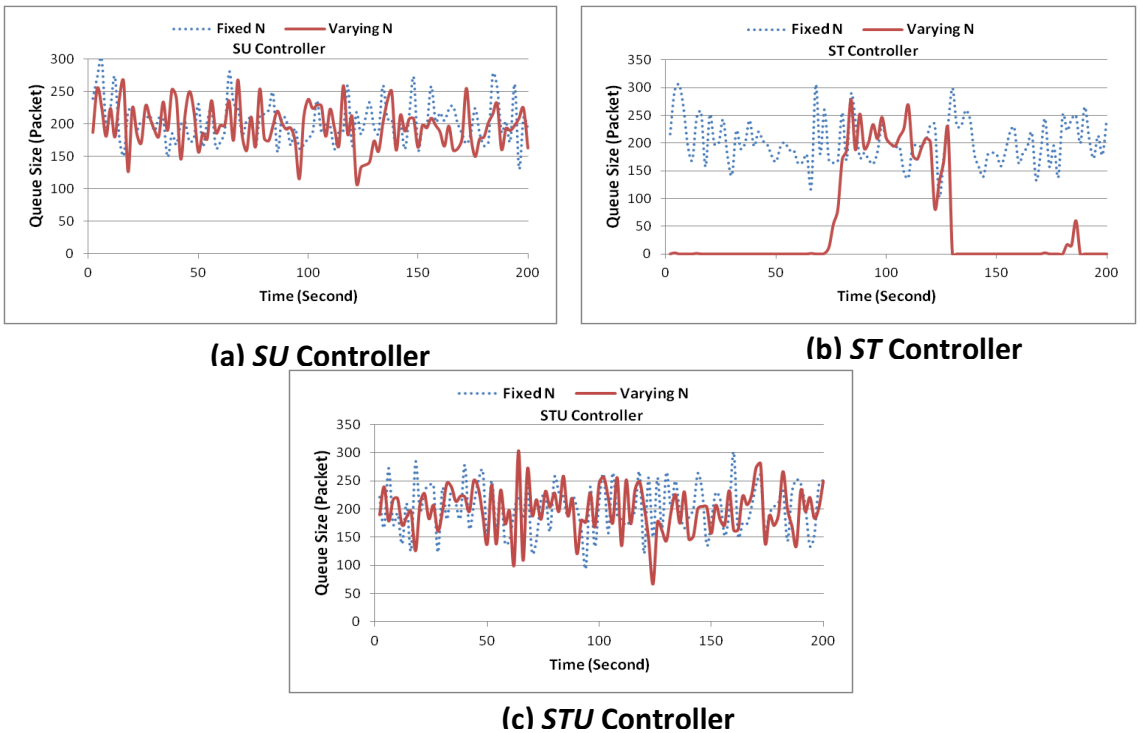
Our controller has been designed for a normal network load of *N*=100. Since a source may not be active (sending data to the network) all the time, we would like to investigate the robustness of the designed H_2/H_∞ controllers when the number of TCP connections is varying w.r.t. the nominal value of 100.

Table 4.15: Sources Characteristic in Varying Network Load Scenario

Node#	Start Time	Stop Time
ftp 1-20	40s	200s
ftp 21-50	0	120
ftp 50-60	0	160
ftp 61-100	0	200
ftp 101-120	80	120

a) Varying Network Load

Table 4.15 gives the start and stop times of different ftp sources so that the total number of connections is varying. According to the table, there are 80 ftp sources sending data in the time interval of (0s, 40s), 100 in the interval of (40s, 80s), 120 in the interval of (80s, 120s), 70 in the time interval of (120s, 160s) and 60 in the time interval of (160s, 200s).



(a) *SU* Controller

(b) *ST* Controller

(c) *STU* Controller

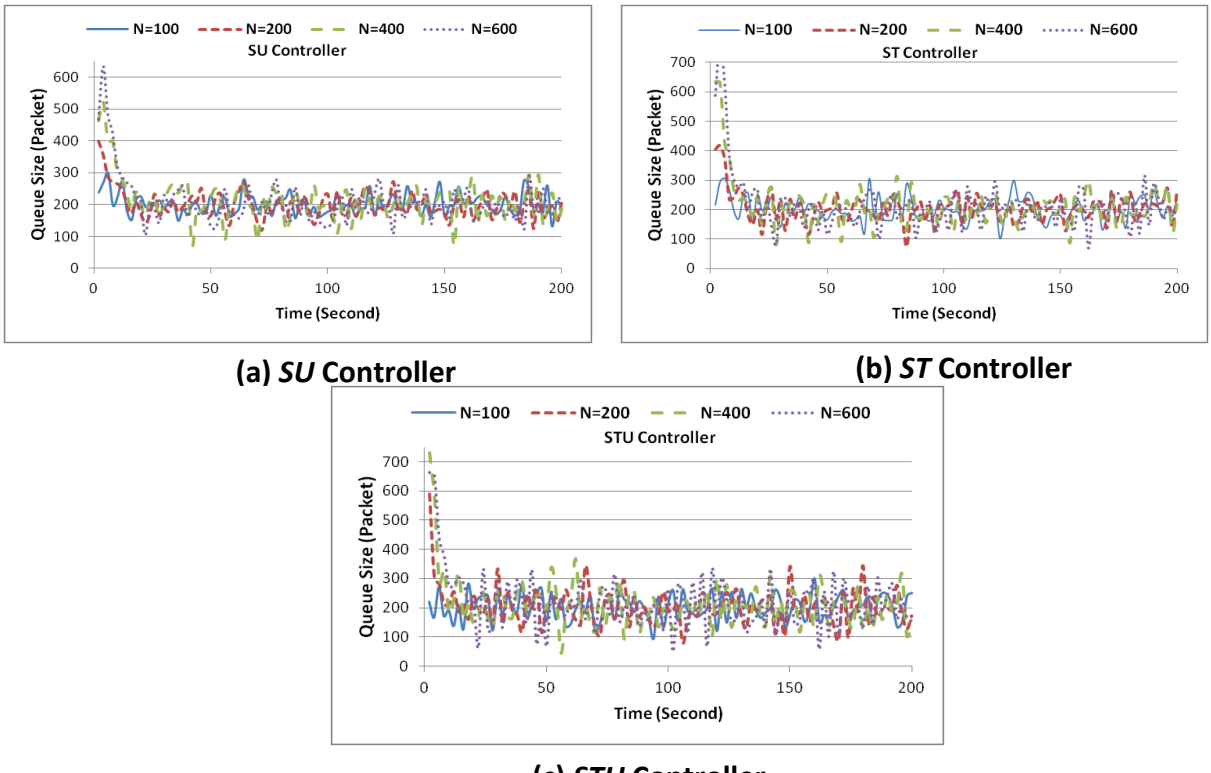
Fig. 4.25: Queue Size of the SU, ST and STU Controllers Under Varying Network Load Scenario

Fig. 4.25 compares the queue size for the designed H_2/H_∞ controller when the number of ftp sources is fixed at the nominal value of N=100, and when the number varies. Both the

designed SU and STU controllers performed reasonably well; they are able to reach the desired queue size within a small rise/fall time (~6 sec for the SU controller and 5sec for the STU controller). The queue size for these two controllers has similar oscillations around the desire queue size under the fixed *N* scenario. On the other hand, the ST controller can only work properly between t=80s to 120s when *N*=100. This is because the ST controller does not have weight function $W_U(s)$ to gauge the input sensitivity function *U(s)*, and becomes more sensitive to the changes of the network parameters than the ST and STU controllers.

b) Heavier Network Load

We investigate the controller performance using network load heavier than the nominal value of *N*=100. The load ranges from 200 to 600 connections.



(a) *SU* Controller **(b) *ST* Controller**

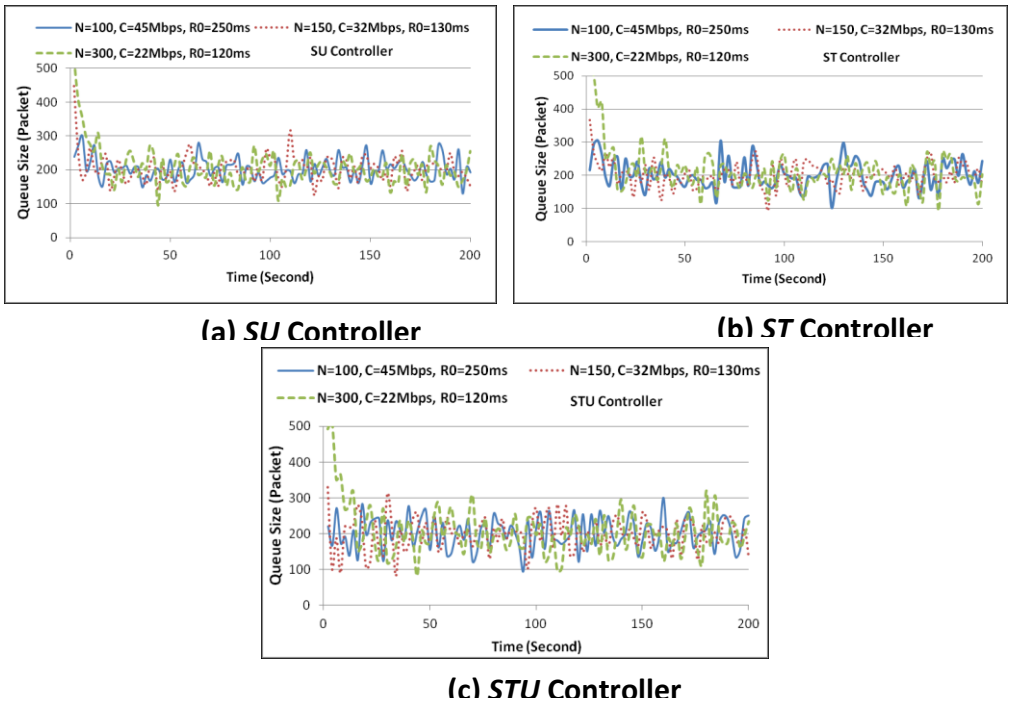
(c) *STU* Controller

Fig. 4.26: Queue Size of the SU, ST and STU Controllers under Heavy Network Load scenario

Fig. 4.26 compares these queue size performances against the nominal value of *N*=100. By having twice the TCP sessions of *N*=200, the peak queue size values are 400, 410, 600 and

rise/fall times are 9s, 10s, 9s for the SU, ST and STU controller respectively. When *N*=400, the peak queue values are increased to 510, 650 and 660 packets and the rise/fall time are 10s, 10s and 10s. When *N*=600 the peak queue values are 620, 700 and 720 packets and the rise/fall time are 15s, 13s, 22s for the SU, ST and STU controllers respectively.

In summary, all designed controllers have proper performance; they are able to reach to desired queue size within a small rise/fall time. By increasing the number of TCP sessions, the peak queue size (in the beginning) and rise/fall time are also increased.



(a) *SU* Controller (b) *ST* Controller

(c) *STU* Controller

Fig. 4.27: Queue size Comparison of the SU, ST, STU Controllers under Different Network Parameters

4.5.2.10 Different Nominal Values for R_0, C and N

Fig. 4.27 shows the performance of the designed controllers under different combinations of *N, C,* R_0. The solid blue line is the time evolution of the queue length for a system with the nominal values of (*N, C,* R_0) = (100, 45 Mbps, 250ms). This controller system has the smallest rise time T_r (~6s, 9s and 2s for the SU, ST and STU controllers respectively) and peak value (~290, 306, 245 packets for the SU, ST and STU controllers respectively). The controllers with (*N, C,* R_0)=(150, 32Mbps, 130ms) and shown by the dashed red line has the second smallest peak queue value (~450, 370 and 330 packets) at T_r ~6s, 15s and 3s for the SU, ST and STU controllers

respectively. This is followed by the (N, C, R_0) = (300, 22Mbps, 120ms) system with T_r ~15s, 16s and 15s and peak queue value (~500, 500 and 500 packets for the SU, ST and STU controllers respectively).

From the above and others not shown (in order not to clutter the diagram), one can see that all cases are stable similar to the system with the nominal values as they all are able to maintain the queue size to the level of ~200 packets. Furthermore, a larger N leads to higher peak queue value since more sources are sending packet to the network.

4.6 Concluding Remarks

This chapter has proposed the window-based H_2/H_∞ controller to control traffic congestion in the network. The feature of our controllers is the additional H_2 formulation, which improves the transient performance in addition to the robustness feature of H_∞ controller. Our design uses sensitivity and weight functions to monitor the closed-loop dynamics and bounds them to have reasonable performances. We have investigated different combinations of weight functions in the controller design to achieve better performance. The recommendations on the weight functions are summarized in Chapter 7. Our simulation results demonstrate the robustness of these control systems (the SU, ST, and STU controllers) in being able to settle to a desired queue length in the presence of disturbances and network parameters changes. Note that finding the suitable/applicable weight functions for the ST controller is a time consuming procedure. As mentioned in Section 4.3.2, although we have tried wide ranges of parameters values for W_s and W_T, only a few of them result in a reasonable/workable ST controller. By comparison, our proposed H_2/H_∞ controllers can provide significant improvement in transient response (such as rise/fall time and peak queue value) and in steady state such as the small oscillation from the target queue size as well as having lower queueing delay and packet loss rate. These improvements are fundamental ingredients for better QoS design. All the designed controllers have shown robust performance under bandwidth changes, having uncontrolled traffic, different RTT, different N scenarios. For varying network load scenario, the ST controller cannot follow the changes

Chapter 5
Rate-Based H_2/H_∞ Controllers

After designing the window-based H_2/H_∞ router controllers, we now change our direction to (study) the rate-based router controller that can provide relatively smooth source sending rates for streaming media traffic. In this chapter, we shall design the H_2/H_∞ rate-based controller to tackle the network traffic, and then conduct analysis and performance evaluation for fixed network parameter values. We shall derive and describe the plant model in Section 5.1. Section 5.2 is on the design of the H_2/H_∞ rate-based controller. This is followed by the RB-SU (Rate-Based SU) controller design and the controller stability analysis in Section 5.3. Section 5.4 covers the RB-STU (Rate-Based STU) controller design and the controller stability analysis. Performance under different scenarios is evaluated in Section 5.5. Since the development corresponds closely to that of the window-based, we shall skip some parts/discussions if there is no confusion. Instead, we shall make a comparison with the window-based controller at the end.

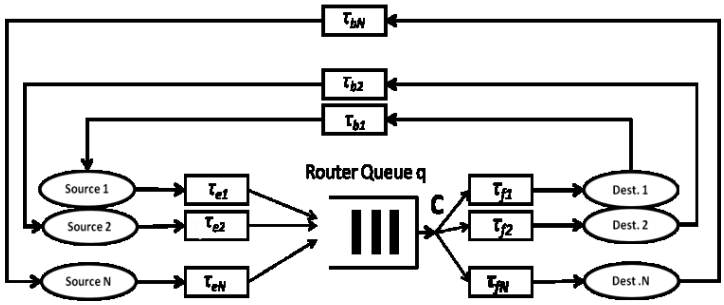


Fig. 5.1: TCP/AQM Plant Model

5.1 Rate-Based Control System Model for the TCP/AQM Traffic

For the convenience of readers, we repeat Fig.2.2 of Ch.2 in Fig. 5.1 above to derive our plant/router model from which one will see the delay component of the TCP/AQM feedback control model of Fig. 2.3 takes on a different form from the window-based controller.

5.1.1 The TCP/AQM Plant Model

Consider the *N* sources sending their internet traffic to their respective destinations in Fig. 5.1 For *i*=1,2,…,*N*, let $u_i\ (t)$ be the current sending rate of source *i*; $u_i'(t)$ be the sending rate of source *i* allowed by the router to be sent back to the sender, and *v(t)* be the incoming aggregate uncontrolled flow rate. Let *C* be the link bandwidth (also called the service rate here) measured in bps (bit per second). For a particular source-destination pair *i*, τ_{ei} is the time delay of a packet from source *i* to the router, and τ_{fi} is the time delay of the packet of source *i* from the router to its destination *i*, while τ_{bi} is the feedback delay from destination *i* back to source *i*. Obviously $\tau_i = \tau_{ei} + \tau_{fi} + \tau_{bi}$ is the RTT (Round Trip Time).

Our proposed rate-based controller in the router can determine a desirable source transmission rate $u'_i\ (t)$ for each flow based on the instantaneous queue length *q(t)* of the buffer in the router. The controller then advertises it to the source through the IP packet and the ACK packet, so that the source can update/regulate its current transmission rate $u_i(t)$ to provide the best-effort service traffic.

The change of queue length in the router buffer is the summation of the input rates of the *N* best-effort service traffic streams and the guaranteed service traffic stream minus the service rate. It can be written as follows.

$$\dot{q}(t) = \sum_{i=1}^{N} u_i(t - \tau_{ei}) + v(t) - C \qquad (5.1)$$

Source *i* will update its current rate $u_i(t)$ according to the calculated rate $u_i'(t)$ sent from the router. Due to the delay in between as shown in Fig. 5.1, we have

$$u_i\ (t) = u'_i\left(t - \tau_{fi} - \tau_{bi}\right) \qquad (5.2)$$

The total delay $\tau_{fi} + \tau_{bi}$ captures all delay components such as the propagation delay, the queuing delay and the processing delay along the path. By substituting eqn. (5.1) into eqn. (5.2), we can obtain the dynamics of the router as

$$\begin{aligned} \dot{q}(t) &= \sum_{i=1}^{N} u_i'\left(t - \tau_{ei} - \tau_{bi} - \tau_{fi}\right) + v(t) - C \\ &= \sum_{i=1}^{N} u_i'(t - \tau_i) + v(t) - C \end{aligned} \qquad (5.3)$$

Therefore, the transfer function between the instantaneous queue length *q(t)* and the advertised source transmission rate $u_i'(t)$ is given by

$$T_{qu'} = \sum_{i=1}^{N} \frac{e^{-s\tau_i}}{s} \quad (5.4)$$

To simplify the controller design, we can approximate the instantaneous round trip time by the average value τ of τ_i, i.e., $\tau = R_0 = \sum_{i=1}^{N} \tau_i / N$. Such plant approximation has been widely adopted in the industrial process control system design e.g., [HoHo03], which also has found its application in network traffic control e.g., [LoAn05]. Therefore, under this approximation, we can update eqn. (5.4) as

$$T_{qu'} = \frac{Ne^{-sR_0}}{s} \quad (5.5)$$

To be compatible with the current dominant TCP/IP protocol in the Internet, we need to convert the advertised source transmission rate into an advertised window size for the controlled source nodes. Similar methods have been well accepted in the analysis of TCP/IP networks e.g., [LoAn05]. Therefore, the advertised window size $win_i'(t)$ for the controlled source node *i* can be calculated as follow.

$$win_i'(t) = \tau_i . u_i'(t) \quad (5.6)$$

By approximating τ_i to the average round trip time R_0, the transfer function between the instantaneous queue length *q(t)* and the current advertised window-size $win'(t)$ (which is also the AQM plant model the H_2/H_∞ controller will be designed for) can be written as following equation.

$$P(s) = T_{qwin'} = \frac{Ne^{-sR_0}}{R_0 s} \quad (5.7)$$

By applying the first order Pade approximation [TuBa10] with $e^{-R_0 s} \approx \frac{1-(R_0/2)s}{1+(R_0/2)s}$, the plant model can be rewritten as

$$P(s) = \frac{N(1-\left(\frac{R_0}{2}\right)s)}{(1+(\frac{R_0}{2})s)R_0 s}. \quad (5.8)$$

The operating point of the system is defined by the set of parameter values that would stabilize the router queue length, i.e., when the queue length change rate $\dot{q}(t) = 0$ in eqn. (5.1). Therefore, the transmission rate, advertised window size operating points (by substituting in eqn. (5.6)) and queue length, (U_0, W_0, q_0) can be determined as follows.

$$U_0 = \frac{C - \nu}{N}$$

$$W_0 = \frac{R_0(C-\nu)}{N} \quad (5.9)$$

$$q_0 = q_{ref}$$

where *v* is uncontrolled traffic rate and q_{ref} is the desired queue size.

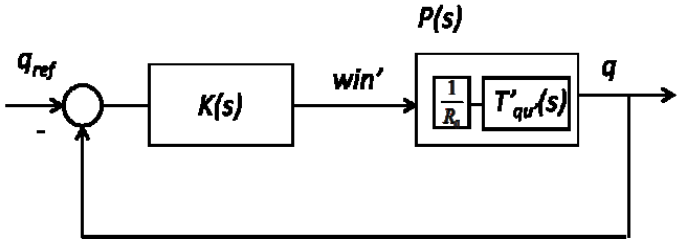


Fig. 5.2: Rate-Based TCP/AQM Feedback Control System

5.1.2 Feedback Control Model of the H_2/H_∞ Controller

Fig. 5.2 shows a rate-based AQM feedback control model from which we can build our H_2/H_∞ control in Fig. 2.4 of Chapter 2. Here, *K(s)* is the controller and *P(s)* is the plant model. The AQM controller input is the deviation of the queue length (q-q_{ref}) from the reference point, and the output is the advertised window size *win'* of a source. The plant model has two parts: 1) the transfer function between the queue length and the advertised rate; 2) the gain of $1/R_0$ to make the rate- based system compatible to the window-based TCP system. The plant model above can now be used to design the controller using the SU, ST and STU approaches given in Chapter 3.

With respect to the general H_2/H_∞ control configuration given in Fig 2.4 and the control system model in Fig. 5.2, the weighted advertised window size *win'* is the exogenous output z_1 in Fig. 3.1, and the weighted queue length q is the exogenous output z_2. Indeed, the queue size error is also the controller input $\boldsymbol{v}$ in Fig 2.4 and the advertised window size *win'* is the control signal $\boldsymbol{u}$. In addition, y_{ref} indicates the reference (desired) queue size in the TCP/AQM system.

5.2 Designing the H_2/H_∞ Rate-Based Controller

The design principle has been detailed in Section 3.1 where we want to minimize $\gamma^2 = \alpha\|\boldsymbol{E}_\infty\|_\infty^2 + \beta\|\boldsymbol{E}_2\|_2^2$ where $\|\boldsymbol{E}_\infty\|_\infty$ and $\|\boldsymbol{E_2}\|_2$ capture the steady-state and the transient performance respectively. As part of the design process, one would need to select weights for

the Sensitivity function, the Input Sensitivity function and the Complementary Sensitivity function as described in Section 3.2. Like the window-based ST controller in Section 4.3.2, it is hard to find the suitable/workable weight functions for the RB-ST controller. Also, all the RB-ST controllers we investigated do not appear to have robust performance under different network load scenarios. Therefore, its design and performance evaluation will not be presented in this chapter.

Again, in order to avoid the cumbersome notation, we shall use "RB-SU Controller" to mean "Rate-Based SU H_2/H_∞ Controller. Likewise, we use "RB-ST Controller" and "RB-STU Controller" with the "H_2/H_∞" portion understood. Unless there is confusion, we shall also omit the "frequency" variable s in the functions.

5.2.1 RB-STU Example on Weight Selection

Parallel to the window-based case in Chapter 4, we provide here an example of selecting the weights for the RB-STU Controller. Readers can find the weight selection of the RB-STU controller in Appendix I. The same network characteristic for the TCP/AQM network from Section 4.2.1.1 shall be considered here where *N*= 100 greedy ftp sources along with 20 http sources, round trip time R_0=0.25s, a T3 link bandwidth of *C*=9708 packets/s with an average packet size of 576 bytes and a buffer size of 1000 packets. Using these parameter values, the network plant model can be described as $P(s) = \frac{100e^{-0.25s}}{0.25s}$. By using the first order Pade approximation $e^{-R_0 s} = \frac{1-(R_0/2)s}{1+(R_0/2)s}$ to convert the infinite dimensional system to a rational transfer function, the plant model can be rewritten as

$$P(s) = \frac{100(1-\left(\frac{0.25}{2}\right)s)}{0.25s(1+\left(\frac{0.25}{2}\right)s)} = \frac{400-50s}{0.125s^2+s}. \qquad (5.10)$$

After performing several simulations to find the $W_s(s)$, $W_U(t)$ and $W_T(s)$ similar to the process of the window-based controller in Chapter 4, the weight functions are chosen as following:

$$W_s(s) = \frac{\frac{s}{5}+4.5}{s+4.5\times 10^{-3}},\ W_u(s) = 10,\ \ W_T(t) = \frac{0.001+3s}{1+0.1s}$$

By solving the optimization design problem of eqn. (3.3) using the *Robust Control*-toolbox in

MATLAB with [α , β] =[1, 1] and [δ_0 ν_0] = [200, 200], the RB-STU controller is obtained as

$$K(s) = \frac{3.361 \times 10^{-5}\,s^3 + 0.0006015s^2 + 0.002696\,s + 1.99 \times 10^{-5}}{7.34 \times 10^{-5}\,s^4 + 0.004836s^3 + 0.1504s^2 + 0.9886s + 0.004443}$$

In the designed controller, the H_∞-norm of the closed-loop transfer functions from ***w*** to Z_∞ is determined to be $\|E_\infty\|_\infty = 4.145$ and the H_2-norm of closed-loop transfer functions from ***w*** to Z_2 is $\|E_2\|_2 = 0.5069$. These norms give the minimum of the mixed H_2/H_∞ norm at $\gamma_{min} = 4.1759$.

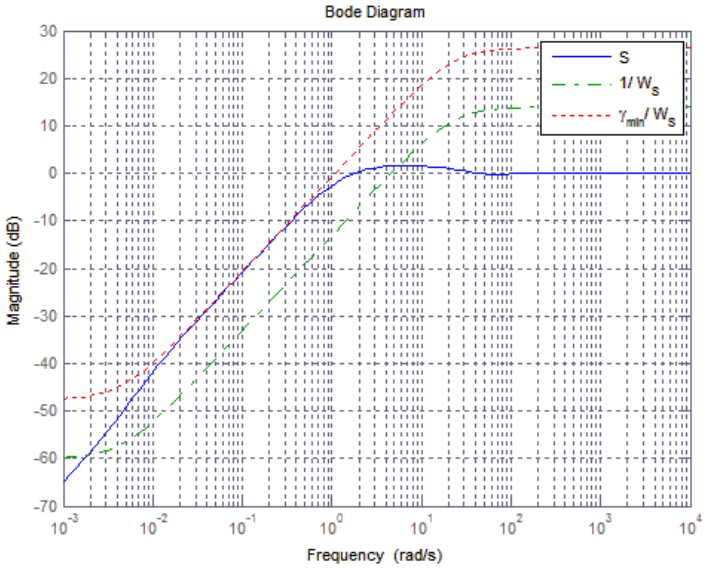


Fig 5.3: Bode Diagram of *S*, $1/W_S$ and γ_{min}/W_S for the RB-STU Controller

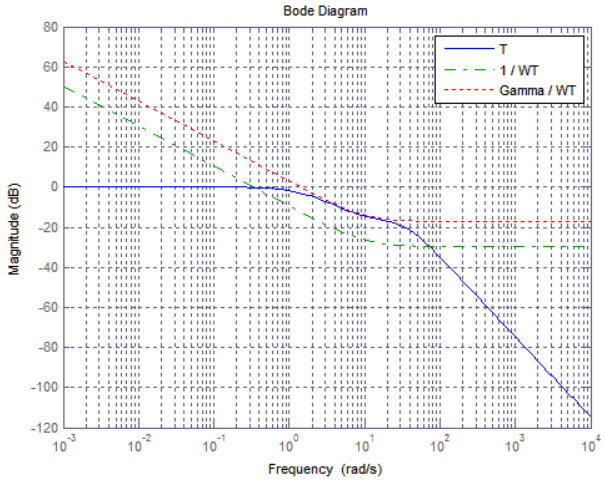


Fig 5.4: Bode Diagram of *T*, $1/W_T$ and γ_{min}/W_T for the RB-STU Controller

In the interest of investigating the feasibility of the proposed $W_s(s)$, Fig 5.3 presents the Bode diagrams of the sensitivity function $S(s)$ (indicated by the solid blue curve), the inverse of the sensitivity weight function $1/W_S$ (by the dotted red curve) and γ_{min}/W_S (by the dashed green curve). The sensitivity of the system output to disturbance and the error between system output and the reference input is small at low frequencies due to small sensitivity function $S(s)$ at low frequencies. On the other hand, the magnitude of $S(s)$ attains the asymptotic value 1 (0dB) at high frequencies which demonstrates that the noise and influence on the tracking error should be attenuated [DoBi10]. Furthermore, Fig 5.3 has verified that $|S(j\omega)|$ is upper bounded by $\gamma_{min}/|W_S|$ which is one of the requirements in designing RB-STU controllers; please see section 3.1.2.1 for more details.

Fig 5.4 presents the Bode diagrams of the complementary sensitivity function T, the inverse of the complementary sensitivity weight function $1/W_T$ and γ_{min}/W_T. It is shown that T is 0 dB at low frequencies, and drop off at high frequencies, which reduces the sensitivity to noise and uncertainty at high frequencies. So far, we have mentioned that, $|T|$ is upper bounded by $1/|W_T|$. Fig. 5.4 actually verifies that $|T|$ is upper bounded by $\gamma_{min}/|W_T|$.

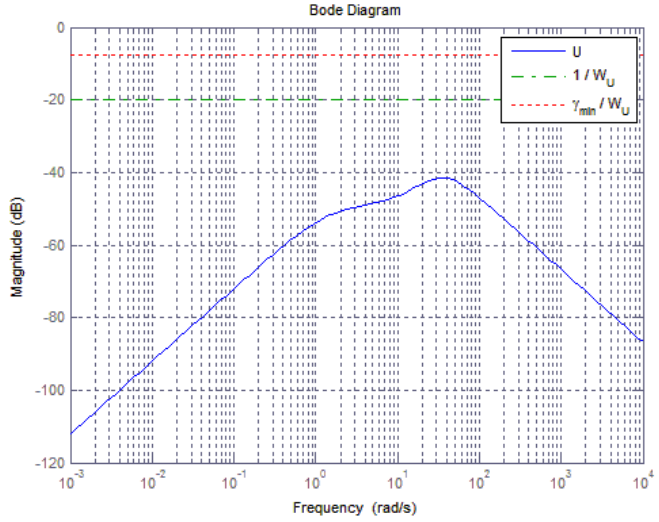


Fig 5.5: Bode Diagram of U, $1/W_U$ and γ_{min}/W_U for the RB-STU Controller

Comparably with other sensitivity functions, Fig. 5.5 represents the Bode diagrams of the input sensitivity function, $U(s)$, the inverse of the sensitivity weight function $1/W_U$ and γ_{min}/W_U. This figure investigates that an appropriate $W_u(s)$ has been selected while $|U(j\omega)|$ is upper

bounded by $\gamma_{min}/|W_U|$.

5.3 The RB-SU Controller

The RB-SU controller is a rate-based controller using the H_2/H_∞ approach mentioned in Chapter 3. It is designed with the desire of improving the transient performance of the SU system in Chapter 4 along with the robustness of the system. It can achieve relatively small fluctuations in the router queue size. The source rates and the corresponding window sizes are calculated based on the instantaneous router queue size and will be fed back to sources through Acknowledgement packets.

5.3.1 Controller Design

Similar to the SU controller, the generalized plant model for the RB-SU controller contains two weight functions: the sensitivity $W_s(s)$ and the input sensitivity weight functions $W_u(s)$. We shall use the same weight function equations mentioned in Section 3.1.2 to design our RB-SU controller.

5.3.1.1 Design Example

The same TCP/AQM network system parameter values to those Chapter 3 and Chapter 4 and Section 5.2.1 are used in this section. Therefore, the plant model is the same as eqn. (5.10). We also use *Robust Control*-toolbox in MATLAB by assigning the same weights of [α, β] = [1, 1] to $\|\boldsymbol{E}_\infty\|_\infty$ and$\|\boldsymbol{E_2}\|_2$. Using the procedure in Fig. 3.1 and choosing some relatively large numbers for δ_0 , ν_0 (e.g., $[\delta_0, \nu_0]$ = [200, 200]) in order for the iteration process converge to the optimum result. After some iterations (about 10), the weight functions $W_U(s)$ and $W_s(s)$ are chosen as follows.

$$W_s(s) = \frac{\frac{s}{5}+4.5}{s+4.5\times10^{-3}}, W_u(s) = 10. \qquad (5.11)$$

The designed RB-SU controller for the plant model described in (5.10) using weight functions mentioned in (5.11) is as following:

$$K(s) = \frac{0.0009308\,s^2 + 0.00747\,s + 0.000198}{0.002723\,s^3 + 0.06372\,s^2 + 0.998\,s + 0.004442} \quad . \qquad (5.12)$$

From the designed controller, we obtain the H_∞-norm for the closed-loop transfer functions from $\boldsymbol{w}$ to $\boldsymbol{Z_\infty}$ as $\|\boldsymbol{E}_\infty\|_\infty = 1.5545$, and the H_2-norm for the closed-loop transfer functions from $\boldsymbol{w}$ to $\boldsymbol{Z_2}$ as $\|\boldsymbol{E_2}\|_2 = 0.8203$. Using these norm values, the mixed H_2/H_∞ norm (which is minimized through solving the optimization problem) is $\gamma_{min} = 1.7577$.

Similar to Section 3.4, the feasibility of the designed controller has investigated using Bode diagrams. For more information, please see Appendix G.

5.3.2 Effect of the Design Parameters

Similar to the SU controller in section 4.2 simulations have been executed to assess the effect of the parameters on the performance and robustness of the controller. Specifically, different values of W_U, ω_B, M and A are investigated in the interest of observing their impacts on the system performance. Finally, we make some recommendations on the parameters selection ranges of the weight functions. We do this by varying one weight function (say W_U) and fixing the other weight function (say W_S). Recall that in our simulation to determine the controller design, we assume that there is no uncontrolled traffic, i.e., v=0.

5.3.2.1 Effect of Varying W_S

Like Section 4.2.2, we fix W_u, while varying the parameters of $W_s(s) = \frac{s/M+\omega_B}{s+\omega_B A}$ one at a time. Also see Section 3.2 for the definitions of the parameters.

a) Varying M

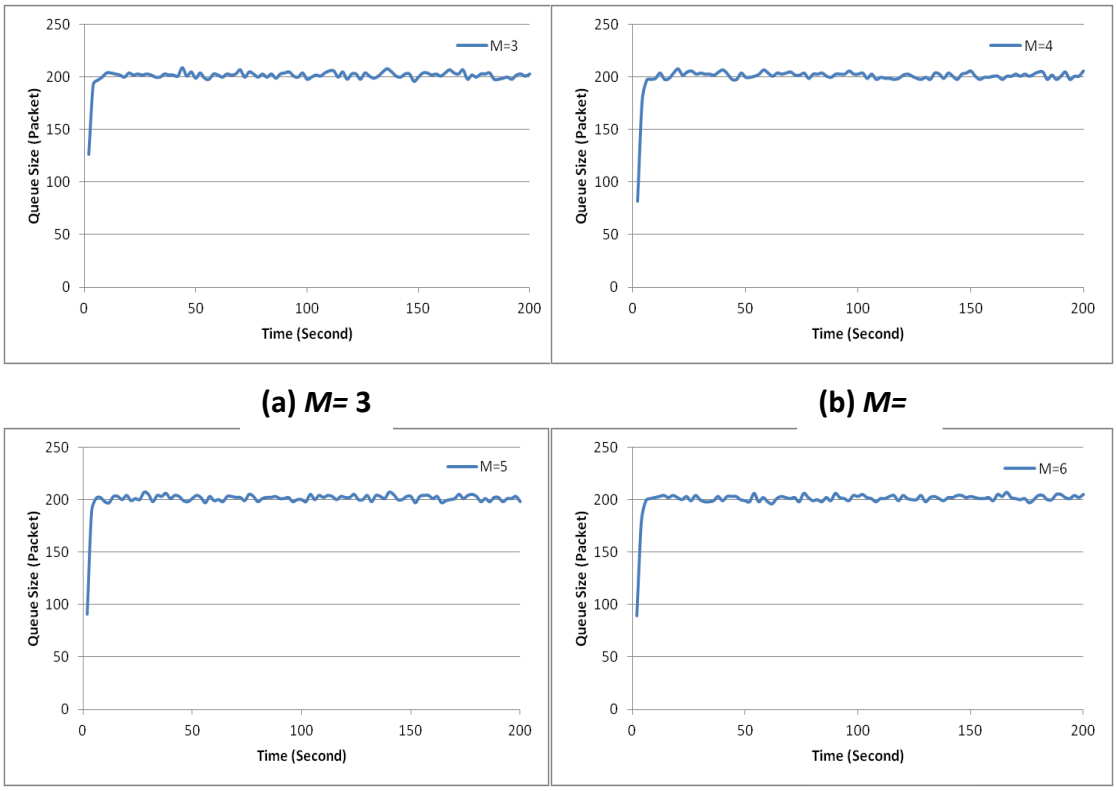
To find the best M, we vary M while keeping fix W_U, ω_B and A at 10, 4.5 and 10^{-3} respectively.

Table 5.1: Different RB-SU Controllers Specifications by Varying *M*

W_U=10, ω_B =4.5, A=10^{-3}	γ_{min}	$\|E_\infty\|_\infty$	$\|E_2\|_2$	$e(\infty)$	PM, GM
M=3	1.7504	1.5471	0.8189	9.88×10^{-15}	56.3 °, 6.1dB
M=4	1.7577	1.5545	0.8203	8.88×10^{-16}	55.5 °, 6.0dB
M=5	1.7436	1.5378	0.8218	1.59×10^{-14}	55.0 °, 5.9dB
M=6	1.7358	1.5278	0.8240	6.30×10^{-14}	54.8 °, 5.8dB

Table 5.1 shows different controller specifications including γ_{min} (the minimum mixed

H_2 /H_∞-norm), $\|E_\infty\|_\infty$ (the H_∞-norm of closed-loop transfer functions from ***w*** to ***z***), $\|E_2\|_2$ (the H_2-norm of closed-loop transfer functions from ***w*** to ***z***), the steady state error $e(\infty)$, the phase margin PM, and the gain margin GM as a function of *M*. By increasing *M*, one can see that $\|\boldsymbol{E}_\infty\|_\infty$, $\|\boldsymbol{E}_2\|_2$ and their corresponding γ_{min} decrease (though not significantly) and lead to a more robust controller. Steady state error is increasing as *M* increases. The controllers are stable due to their positive phase margin and gain margin.



(a) *M*= 3 **(b) *M*=**

(c) *M*= 5 **(d) *M*= 6**

Fig. 5.6: Instantaneous Queue Size of the RB-SU Controllers with W_U =10, *A*= 10^{-3} and ω_B=4.5 but Different *M* Values

Table 5.2: Performance of RB-SU Controllers with Different *M*

W_U=10, ω_B =4.5, *A*=10^{-3}	T_r (Second)	Max/Min (Packet)	μ_q (Packet)	σ_q(Packet)
M=3	6	209/126	201	8
M=4	7	202/88	200	12
M=5	4	207/91	200	11
M=6	5	207/89	200	11

Fig. 5.6 shows the instantaneous queue size for the RB-SU controller when simulated in OPNET for *M* ranging from *M*=3 to 6. One can see that all queue sizes are relatively similar. They all reach steady state in a short time and have remarkably small variations. In order to have a deeper insight, Table 5.2 provides different performance measures including T_r (rise/fall time), maximum and minimum packet size value, μ_q (mean queue length) and σ_q (queue length standard deviation). One can see that, T_r, max/min, μ_q, σ_q are not changing monotonically w.r.t. *M*. The smallest T_r belongs to M=5, while the smallest σ_q (i.e. the least variation) belongs to M=3. All the queue sizes have relatively the same μ_q (i.e. 200 packets) which is the desired queue size value except for *M*=3 at μ_q=201. Since all the queue size have similar results, *M*=5 is chosen due to its smallest T_r.

Table 5.3: Different RB-SU Controllers Specifications by Varying *A*

W_U=10, ω_B=4.5, *M*=5	γ_{min}	$\|E_\infty\|_\infty$	$\|E_2\|_2$	$e(\infty)$	PM, GM
A=10^{-2}	1.7423	1.5422	0.8106	1.10×10^{-13}	56.3°, 6.0dB
A=10^{-3}	1.7577	1.5545	0.8203	8.88×10^{-16}	55.5°, 6.0dB
A=10^{-4}	1.7593	1.5560	0.8209	2.02×10^{-14}	55.4°, 5.9dB
A=10^{-5}	1.7457	1.5418	0.8188	1.73×10^{-14}	55.0°, 5.9dB

b) Varying *A*

In this section, we observe the effect of *A* by setting *A* to different values while fixing W_U, ω_B and *M* at 10, 4.5 and 5 respectively.

Table 5.3 shows different designed controllers specifications including γ_{min}, $\|\boldsymbol{E}_\infty\|_\infty$, $\|\boldsymbol{E_2}\|_2$, $e(\infty)$, PM and GM. By increasing *A*, $\|\boldsymbol{E}_\infty\|_\infty$ and $\|\boldsymbol{E_2}\|_2$ and therefore γ_{min} are increasing but not significantly. There is no monotonic increase in steady state error. The largest error occurs at *A*=10^{-2} and the smallest at *A*=10^{-4}. All controllers are stable due to their positive PM and GM values.

Fig. 5.7 illustrates the instantaneous queue size for *A* ranging from *A*=10^{-2} to 10^{-5}. One sees that the system is sensitive to changing *A*. The system cannot reach the desired queue size of 200 packets for *A*=10^{-2} and 10^{-5}. For *A*=10^{-4}, there is a large fluctuation between 1 and 1000 in the transient state before settling down at t=45s. The system is unstable for *A*= 10^{-5} when the buffer is full all the time. The A=10^{-3} appears to be the only case that 200 packets can be

reached.

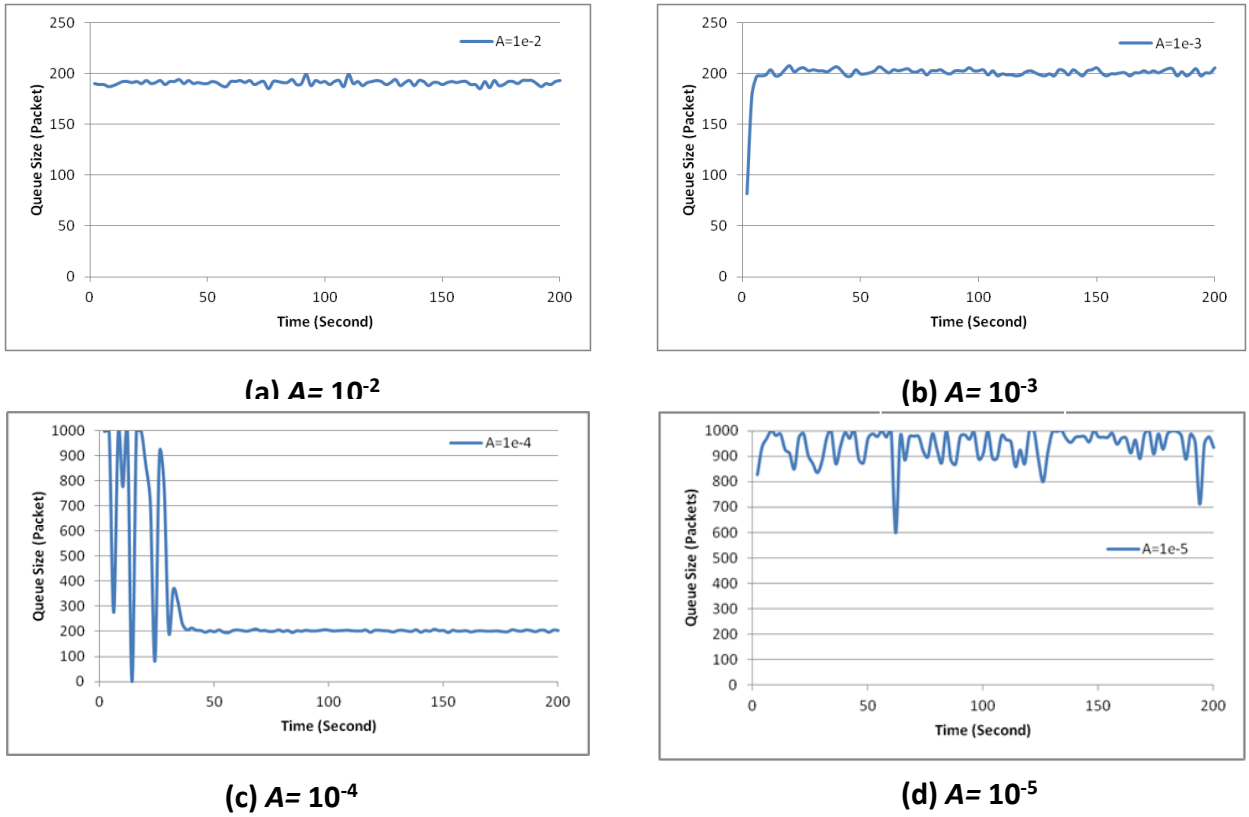


Fig.5.7: Instantaneous Queue Size of the RB-SU Controllers with W_U =10 , ω_B=4.5 and M=5 but Different A Values

Table 5.4: Performance of RB-SU Controllers with Different A

W_U=10, ω_B =4.5, M=5	T_r (Second)	Max/Min (Packet)	μ_q (Packet)	σ_q(Packet)
A=10^{-2}	NA	199/185	190	2
A=10^{-3}	7	202/88	200	12
A=10^{-4}	45	1000/1	282	226
A=10^{-5}	NA	1000/600	941	64

Table 5.4 shows different queueing performance measures including T_r, max/min, μ_q and σ_q as a function of A while fixing W_U, ω_B and M. For A=10^{-2}, μ_q is 190 packets and cannot reach the desire value of 200. For 10^{-3}, the system has a reasonable queue size, a small T_r, acceptable max/min queue sizes (at 202/88) and desirable μ_q and σ_q. Therefore, A=10^{-3} will be chosen for

the final controller. Note that the T_r for both A=10^{-2} and A=10^{-5} are immeasurable as seen in Fig. 5.7 and therefore indicated by NA in the table.

c) Varying ω_B

In this section, we shall vary ω_B while fixing W_U, M and A at 10, 5 and 10^{-3} respectively.

Table 5.5: Different RB-SU Controllers Specifications by Varying ω_B

W_U=10, A=10^{-3}, M=5	γ_{min}	$\|E\|_\infty$	$\|E_2\|_2$	$e(\infty)$	PM, GM
ω_B =2	0.9965	0.8793	0.4688	1.89×10^{-14}	58.4°, 7.4dB
ω_B =4	1.6134	1.4255	0.7555	7.99×10^{-14}	55.8°, 6.1dB
ω_B =4.5	1.7577	1.5545	0.8203	8.88×10^{-16}	55.5°, 6.0dB
ω_B =5	1.8993	1.6814	0.8832	1.44×10^{-15}	55.1°, 5.8dB
ω_B =6	2.1766	1.9303	1.0056	1.21×10^{-14}	54.6°, 5.5dB
ω_B =8	2.7046	2.4077	1.2320	4.64×10^{-14}	53.8°, 5.2dB

Table 5.5 compares controllers of different specifications (by varying ω_B) in term of their γ_{min} , $\|\boldsymbol{E}_\infty\|_\infty$, $\|\boldsymbol{E_2}\|_2$, $e(\infty)$, PM, and GM. When ω_B is increasing, $\|\boldsymbol{E}_\infty\|_\infty$, $\|\boldsymbol{E_2}\|_2$ and consequently γ_{min} increases which results in a less robust controller. Steady state error does not show monotonic changes w.r.t. increasing ω_B. The smallest and the largest $e(\infty)$ occurs at ω_B=4.5 and ω_B=4.5 respectively. Furthermore, the positive PM and GM values indicate that the controllers are stable.

Fig. 5.8 below shows the instantaneous queue size for the RB-SU controller for ω_B ranging from ω_B=2 and 8. One can see that they all reach their steady state in a short time and have small variations. For the controllers using ω_B =2 and 4, unlike other controllers, their queue sizes start from a value larger than 200.

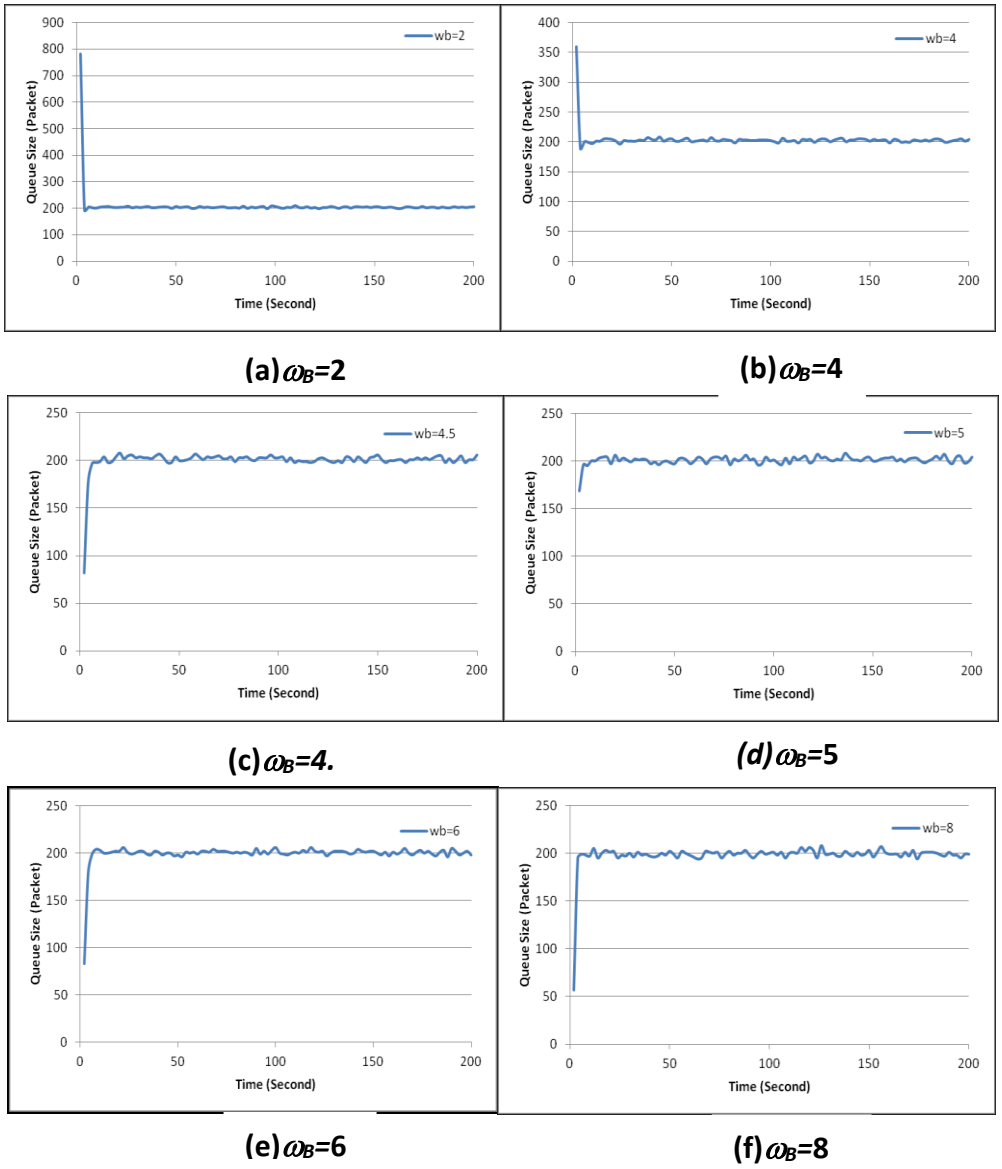


Fig. 5.8: Instantaneous Queue Size of the RB-SU Controllers with W_U =10, A=10^{-3} and M=5 but Different ω_B Values

In order to have a deeper insight, Table 5.6 below provides different performance measures including T_r (rise/fall time), the maximum and minimum packet size values, μ_q (the mean queue length) and σ_q (queue length standard deviation). One can see that, T_r, max/min, μ_q, σ_q are not changing monotonically w.r.t. ω_B. The smallest T_r (~3s) belongs to ω_B =3 while the longest T_r (~7s) belongs to ω_B =4.5 and 5. The smallest peak queue value belongs (~202 packets) to ω_B =4.5 while the smallest μ_q belongs to ω_B =4.5 and 5. The largest one (~360 packets) belongs to ω_B =4. Moreover, the two smallest σ_q belong to ω_B =5 (~4 packets) and ω_B =4.5 and 6

(~12packets). For the final controller, we chose ω_B =4.5 which gives mean queue equal to desired queue size value and the least maximum queue length with reasonable σ_q.

Table 5.6: Performance of RB-SU Controllers with Different ω_B

W_U=10, A=10^{-3}, M=5	T_r (Second)	Max/Min(Packet)	μ_q (Packet)	σ_q(Packet)
ω_B =2	5	782/192	208	57
ω_B =4	3	360/189	203	15
ω_B =4.5	7	202/88	200	12
ω_B =5	7	208/169	200	4
ω_B =6	5	206/83	199	12
ω_B =8	6	208/57	197	14

According to the simulation results presented in Sections 5.3.2.1.1 to 5.3.2.1.3, we shall choose M=5, A=10^{-3} and ω_B=4.5 and therefore $W_s = \frac{s/5+4.5}{s+4.5\times10^{-3}}$ to design the RB-SU controller.

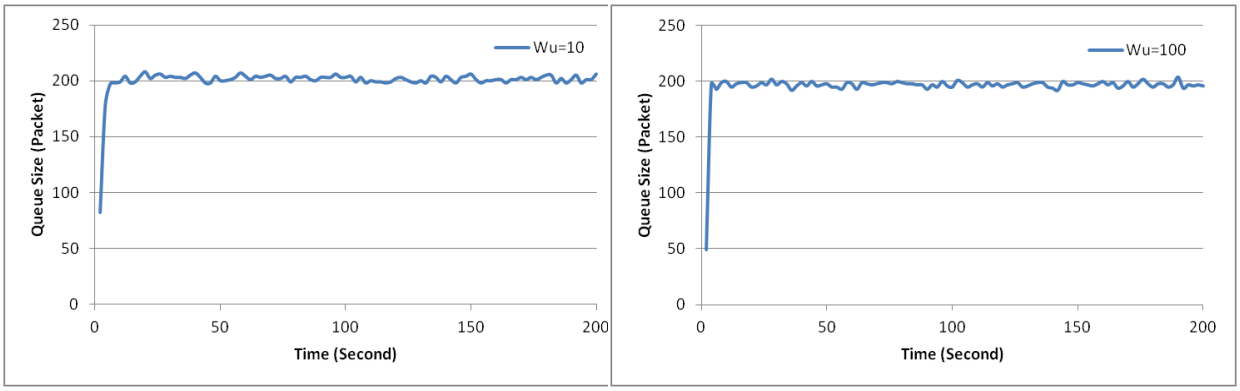
5.3.2.2 Effect of Varying W_U

We now fix $W_s(s)$, and vary the weight function W_u to observe the effect of W_u on the RB-SU controller performance.

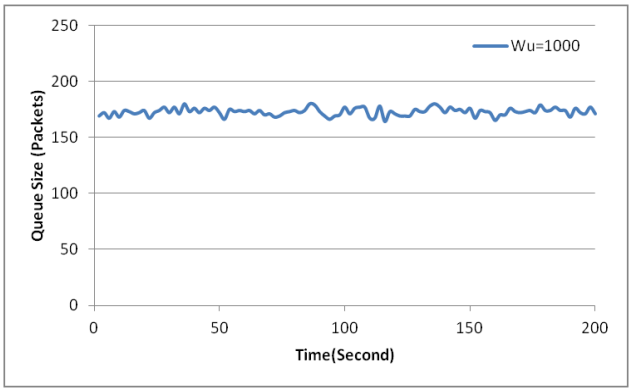
Table 5.7: Different RB-SU Controllers Specifications by Varying W_u

M=5, A =10^{-3}, ω_B =4.5	γ_{min}	$\|E_\infty\|_\infty$	$\|E_2\|_2$	$e(\infty)$	PM, GM
W_U=10	1.7577	1.5545	0.8203	8.88×10^{-16}	55.5°, 6.0dB
W_U=10^2	3.1644	2.6568	1.7190	5.63×10^{-14}	59.0°, 9.0dB
W_U=10^3	6.6469	5.3548	3.9379	1.60×10^{-13}	61.9°, 13.8dB

Table 5.7 tabulates the performance of $\boldsymbol{\gamma_{min}}$, $\|\boldsymbol{E}_\infty\|_\infty$, $\|\boldsymbol{E_2}\|_2$, $\boldsymbol{e}(\infty)$, PM and GM as a function of W_u while fixing W_s with M=5, A=10^{-3} *and* ω_B=4.5. By increasing W_u, the controller becomes less robust due to larger $\boldsymbol{\gamma_{min}}$, $\|\boldsymbol{E}_\infty\|_\infty$ and $\|\boldsymbol{E_2}\|_2$ as well as a larger $\boldsymbol{e}(\infty)$. On the other hand, all the controllers are still stable due to their positive PM and GM values.



(a) W_U =10 **(b) W_U =100**



(c) W_U =1000

Fig. 5.9: Instantaneous Queue Size of the RB-SU Controllers using with ω_B=4.5, A= 10^{-3} and M= 5 but Different W_u Values

Fig. 5.9 shows the instantaneous queue size for the RB-SU controller for W_u ranging from W_u =10 to 10^3. One can see that the instantaneous queue size cannot reach the desired value for W_u= 100 and 1000.

Table 5.8: Performance of RB-SU Controllers with Different W_u

M=5, A =10^{-3}, ω_B =4.5	T_r (Second)	Max/Min Value (Packet)	μ_q (Packet)	σ_q(Packet)
W_U=10	7	202/88	200	12
W_U=10^2	6	202/1	195	14
W_U=10^3	NA	202/2	172	3

In order to have a deeper insight, Table 5.8 provides different performance measures. It shows that the mean queue length is 195 and 172 for W_u=100 and 1000 respectively. By using W_u=10, the queue size reaches 200 packets in 7 seconds and its variation is quite reasonable

with a standard deviation of 12 packets. Therefore, we shall chose W_u=10 for the final RB-SU controller.

5.3.3 Stability Analysis

Similar to Section 4.2.3, we shall provide an analysis on the stability of the proposed RB-SU controller using Lyapunov's Indirect Method under state-space representation. The procedure is similar to that for the SU controller in Chapter4.

5.3.3.1 State-Space Representation

We now proceed to our state-space formulation/derivation of our feedback control system. Note that the AQM plant model presented given in eqn. (5.8) now becomes $P(s) = \frac{N\left(1-\left(\frac{R_0}{2}\right)s\right)}{\left(1+\left(\frac{R_0}{2}\right)s\right)R_0 s} = \frac{-\frac{N}{R_0}s+\frac{2N}{R_0^2}}{s^2+\frac{2s}{R_0}}$. As discussed before, this can be written in the general/standard state-space equation of eqn. (4.9) which in repeated below.

$$\begin{cases} \dot{\boldsymbol{x_P}} = \boldsymbol{A_P x_P} + \boldsymbol{B_P u} \\ \boldsymbol{y} = \boldsymbol{C_P x_P} + \boldsymbol{D_P u} \end{cases}.$$

Using the phase variable canonical space state form [Ogat02], we have

$$\dot{\boldsymbol{x_p}} = \begin{bmatrix} 0 & 1 \\ 0 & -\frac{2}{R_0} \end{bmatrix} x_p + \begin{bmatrix} 0 \\ 1 \end{bmatrix} \boldsymbol{u} \tag{5.13}$$

$$\boldsymbol{y} = \begin{bmatrix} \frac{2N}{R_0^2} & -\frac{N}{R_0} \end{bmatrix} \boldsymbol{x_p} \tag{5.14}$$

From which $\boldsymbol{A_P}$, $\boldsymbol{B_P}$, $\boldsymbol{C_P}$ and $\boldsymbol{D_P}$ could be identified by comparison and inspection.

The state-space representation of the two weight functions $W_s(s)$ and $W_u(s)$ in the RB-SU controller design procedure have been introduced in Section 4.2.3 and repeated here:

$$W_s(s) = \left[\frac{A_{W_s}}{C_{W_s}} \middle| \frac{B_{W_s}}{D_{W_s}}\right] = \left[\frac{-\omega_b A}{\omega_b} \middle| \frac{1}{1/M}\right] \tag{5.15}$$

$$W_u(s) = \left[\frac{A_{W_u}}{C_{W_u}} \middle| \frac{B_{W_u}}{D_{W_u}}\right] = \left[\frac{0}{0} \middle| \frac{0}{W_u}\right] \tag{5.16}$$

According to Theorem 4.2, all eigen-values of the matrix $\boldsymbol{A_{cl}} = \begin{bmatrix} \boldsymbol{A_x} + \boldsymbol{B_2 D_K C_y} & \boldsymbol{B_2 C_K} \\ \boldsymbol{B_K C_y} & \boldsymbol{A_K} \end{bmatrix}$ from eqn.(4.14) must be negative to ensure the stability of the system$\boldsymbol{A_{cl}} = \begin{bmatrix} \boldsymbol{A_x} + \boldsymbol{B_2 D_K C_y} & \boldsymbol{B_2 C_K} \\ \boldsymbol{B_K C_y} & \boldsymbol{A_K} \end{bmatrix}$. The matrices $\boldsymbol{A_x}$, $\boldsymbol{B_2}$, $\boldsymbol{C_y}$, the state $\boldsymbol{x_{cl}}$, the input and the coefficient matrices corresponding to the

general plant matrix can be represented as follows:

$$\boldsymbol{A_x} = \begin{bmatrix} [\boldsymbol{A_P}]_{2\times2} & [0]_{2\times1} \\ -\boldsymbol{C_P}\boldsymbol{B_{W_s}} & \boldsymbol{A_{W_s}} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -\frac{2}{R_0} & 0 \\ -\frac{2N}{R_0^2} & \frac{N}{R_0} & -\omega_b A \end{bmatrix} \tag{5.17}$$

$$B_2 = \begin{bmatrix} [\boldsymbol{B_P}]_{2\times1} \\ -\boldsymbol{B_{W_s}}\boldsymbol{D_P} \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \tag{5.18}$$

$$C_y = [[-\boldsymbol{C_P}]_{1\times2} \quad 0] = \begin{bmatrix} -\frac{2N}{R_0^2} & \frac{N}{R_0} & 0 \end{bmatrix} \tag{5.19}$$

Therefore, $\boldsymbol{A_{cl}}$ can be obtained as follows.

$$\boldsymbol{A_{cl}} = \begin{bmatrix} \boldsymbol{A_x} + \boldsymbol{B_2}\boldsymbol{D_K}\boldsymbol{C_y} & \boldsymbol{B_2}\boldsymbol{C_K} \\ \boldsymbol{B_K}\boldsymbol{C_y} & \boldsymbol{A_K} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ -D_k\frac{2N}{R_0^2} & D_k\frac{N}{R_0} - \frac{2}{R_0} & 0 & C_{K_1} & C_{K_2} & C_{K_3} \\ -\frac{2N}{R_0^2} & \frac{N}{R_0} & -w_b A & 0 & 0 & 0 \\ -\frac{2N}{R_0^2}B_{K_1} & \frac{N}{R_0}B_{K_1} & 0 & A_{K_{11}} & A_{K_{12}} & A_{K_{13}} \\ -\frac{2N}{R_0^2}B_{K_2} & \frac{N}{R_0}B_{K_2} & 0 & A_{K_{21}} & A_{K_{22}} & A_{K_{23}} \\ -\frac{2N}{R_0^2}B_{K_3} & \frac{N}{R_0}B_{K_3} & 0 & A_{K_{31}} & A_{K_{32}} & A_{K_{33}} \end{bmatrix}$$

where $A_{K_{ij}}$ is the i^{th} row and j^{th} column of a matrix $\boldsymbol{A_K}$, B_{K_i} is the i^{th} element of matrix $\boldsymbol{B_K}$ and C_{K_i} the i^{th} element of matrix $\boldsymbol{C_K}$ introduced in eqn. (4.12).

5.3.3.2 Example

As mentioned in Section 5.3.2, the determined weight functions to have the RB-SU controller for the mentioned network parameters value are $W_s = \frac{s/5+4.5}{s+4.5\times10^{-3}}$ and W_u=10. According to eqns. (5.15) and (5.16), state-space representation of the weight functions can be as following:

$$W_s(s) = \left[\frac{-\omega_b A}{\omega_b}\middle|\frac{1}{1/M}\right] = \left[\frac{-4.5\times10^{-3}}{4.5}\middle|\frac{1}{0.2}\right] \tag{5.21}$$

$$W_u(s) = \left[\frac{0}{0}\middle|\frac{0}{W_u}\right] = \left[\frac{0}{0}\middle|\frac{0}{10}\right]$$

For the proposed RB-SU controller in Section 5.2.1, the state space representation is as following

$$K(s) = \left[\frac{\boldsymbol{A_K}}{\boldsymbol{C_K}}\middle|\frac{\boldsymbol{B_K}}{\boldsymbol{D_K}}\right] = \left[\begin{array}{ccc|c} -2.6088 & 0.442223 & 1.742938 & 11.492095 \\ 14.116454 & -2.28321 & -8.63650 & 48.8787160 \\ -125.71382 & 6.394021 & -18.42518 & -255.824062 \\ \hline -0.00028809 & -0.0001794 & -0.0013884 & 0 \end{array}\right] \tag{5.22}$$

Therefore, $\boldsymbol{A_{cl}}$ for N=100, R_0=0.25 second and C=9708 packets/second which the controller is

designed for would be as following.

$$A_{cl} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & -8 & 0 & 0.00028809 & -0.00017943 & -0.00138 \\ -6787.5462 & 848.4432 & -0.0045 & 0 & 0 & 0 \\ -36774.7047 & 4596.8380 & 0 & -2.608873 & 0.44222 & 1.7429 \\ -1.5641189 \times 10^5 & 0.1955148 \times 10^5 & 0 & 14.11645 & -2.28321 & -8.6365 \\ 8.186369 \times 10^5 & -1.023296 \times 10^5 & 0 & -125.71382 & 6.39402 & -18.4251 \end{bmatrix}$$

Then, the corresponding eigen-values are

$\lambda = [-0.0045, -0.02709, -5.6474 \pm +7.6819\text{i}, -7.9784, -12.0169]$

which are the combination of real and complex values with negative real part. Therefore, $\boldsymbol{A_{cl}}$ is exponentially stable. According to Theorem 4.2, controller *K(s)* could be called *stabilizing*.

5.4 The RB-STU Controller

We shall now study the H_2/H_∞ rate-based controller using STU formulations introduced in Section 3.3. Following the network and control model summarized in Section 5.1, we use the same plant model *P(s)* as the RB-SU is used, except now we need all three weight functions $W_s(s)$, $W_U(t)$ and $W_T(s)$ to build the general plant model *G(s)* as outlined in Section 2.4. The weight selection has been discussed as an example in Section 5.2.1.

5.4.1 Design Example

The same network characteristic for the TCP/AQM network as in Section 4.2.1.1 will be used here where we have *N*=100 greedy ftp sources, 20 http sources as bursty traffic, round-trip time R_0=0.25s, link bandwidth *C*=9708 and buffer size of 1000 packets. Accordingly, the plant model can be rewritten as $P(s) = \frac{100(1-\left(\frac{0.25}{2}\right)s)}{0.25s(1+\left(\frac{0.25}{2}\right)s)} = \frac{400-50s}{0.125s^2+s}$.

From the example discussed in Section 5.2.1 and after performing simulations to determine the parameters of the three weight functions, we have arrived at

$$W_s(s) = \frac{s/5+4.5}{s+4.5\times10^{-3}},\ W_u(s) = 10,\ W_T(t) = \frac{0.001+3s}{1+0.1s} \qquad (5.23)$$

By solving the optimization design problem of eqn. (3.3) using the *Robust Control*-toolbox in MATLAB with [α , β] =[1, 1] and [δ_0 ν_0] = [200, 200] the RB-STU controller is obtained as

$$K(s) = \frac{3.361\times10^{-5}\ s^3 + 0.0006015s^2 + 0.002696\ s + 1.99\times10^{-5}}{7.34\times10^{-5}s^4 + 0.004836s^3 + 0.1504s^2 + 0.9886s + 0.004443} \qquad (5.24)$$

In the designed controller, the H_∞-norm of the closed-loop transfer functions from $\boldsymbol{w}$ to $\boldsymbol{Z_\infty}$ is $\|\boldsymbol{E}_\infty\|_\infty = 4.145$, and the H_2-norm of closed-loop transfer functions from $\boldsymbol{w}$ to $\boldsymbol{Z_2}$ is $\|\boldsymbol{E}_2\|_2 = 0.5069$. These norms give the minimum of the mixed H_2/H_∞ norm at $\gamma_{min} = 4.1759$.

5.4.2 Effect of the Design Parameters for the RB-STU Controller

Similar to the RB-SU controller in section 5.3.2, simulations have been executed to assess the effect of the parameters on the performance and robustness of the RB-STU controller. Since this exercise repeats the procedure of the RB-SU, SU and other controllers before, we propose to put the detailed investigation of the weight functions and their effects on performances for the RB-STU controller in the Appendix I. Specifically, Appendix I investigates the effect of weight function of W_T while other weight functions are fixed. Specifically, different values of a_1, a_2, a_3, are investigated in the interest of observing their impacts on the system performance while we are using the same W_s and W_u as determined in Section 5.3.1.1 for the RB-SU controller. Then, we will provide some extra examples with other parameters value for W_s and W_u in Appendix I as well as to show the effectiveness of our determined W_s and W_u. Then, the determined weight functions can be stated as follows.

$$W_s(s) = \frac{s/5 + 4.5}{s + 4.5 \times 10^{-3}}, \quad W_u(s) = 10, \quad W_T(t) = \frac{0.001 + 3s}{1 + 0.1s}$$

5.4.3 Stability Analysis

We shall provide a stability analysis of the proposed RB-STU controller in the state-space domain using Lyapunov's Indirect Method. Similar to the controllers designed previously in Chapter 4 and Section 5.3, and using the example stated in this section, the weight functions can be expressed as the transfer functions in terms of the state-space matrices

$$\begin{aligned} W_s(s) &= \left[\frac{A_{W_s}}{C_{W_s}} \middle| \frac{B_{W_s}}{D_{W_s}}\right] = \left[\frac{-\omega_b A}{\omega_b} \middle| \frac{1}{1/M}\right] \\ W_T(s) &= \left[\frac{A_{W_T}}{C_{W_T}} \middle| \frac{B_{W_T}}{D_{W_T}}\right] = \left[\frac{-1/a_3}{a_2/a_3} \middle| \frac{1}{a_1/a_3}\right] \\ W_u(s) &= \left[\frac{A_{W_u}}{C_{W_u}} \middle| \frac{B_{W_u}}{D_{W_u}}\right] = \left[\frac{0}{0} \middle| \frac{0}{W_u}\right] \end{aligned} \tag{5.25}$$

The matrices $\boldsymbol{A_x}$, $\boldsymbol{B_2}$, $\boldsymbol{C_y}$ associated with the generalized plant model in eqns. (4.11) and (4.29)

can be obtained as follow.

$$A_x = \begin{bmatrix} [A_P]_{2\times2} & [0]_{2\times2} \\ -C_P \begin{bmatrix} B_{W_S} \\ B_{W_T} \end{bmatrix} & \begin{bmatrix} A_{W_S} & 0 \\ 0 & A_{W_T} \end{bmatrix} \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & -\frac{2}{R_0} & 0 & 0 \\ -\frac{2N}{R_0^2} & \frac{N}{R_0} & -w_b A & 0 \\ -\frac{2N}{R_0^2} & \frac{N}{R_0} & 0 & -\frac{1}{a_3} \end{bmatrix} \quad (5.26)$$

$$B_2 = \begin{bmatrix} [B_P]_{2\times1} \\ -B_{W_S} D_P \\ B_{W_T} D_P \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$

$$C_y = [[-C_P]_{1\times2} \quad 0 \quad 0] = [\frac{2N}{R_0^2} \quad -\frac{N}{R_0} \quad 0 \quad 0]$$

Then by substituting in eqn. (4.14), one can obtain

$$\boldsymbol{A_{cl}} = \begin{bmatrix} \boldsymbol{A_x + B_2 D_K C_y} & \boldsymbol{B_2 C_K} \\ \boldsymbol{B_K C_y} & \boldsymbol{A_K} \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -D_k \frac{2N}{R_0^2} & D_k \frac{N}{R_0} - \frac{2}{R_0} & 0 & 0 & C_{K_1} & C_{K_2} & C_{K_3} & C_{K_4} \\ -\frac{2N}{R_0^2} & \frac{N}{R_0} & -\omega_b A & 0 & 0 & 0 & 0 & 0 \\ -\frac{2N}{R_0^2} & \frac{N}{R_0} & 0 & -\frac{1}{a_3} & 0 & 0 & 0 & 0 \\ -\frac{2N}{R_0^2} B_{K_1} & \frac{N}{R_0} B_{K_1} & 0 & 0 & A_{K_{11}} & A_{K_{12}} & A_{K_{13}} & A_{K_{14}} \\ -\frac{2N}{R_0^2} B_{K_2} & \frac{N}{R_0} B_{K_2} & 0 & 0 & A_{K_{21}} & A_{K_{22}} & A_{K_{23}} & A_{K_{24}} \\ -\frac{2N}{R_0^2} B_{K_3} & \frac{N}{R_0} B_{K_3} & 0 & 0 & A_{K_{31}} & A_{K_{32}} & A_{K_{33}} & A_{K_{34}} \\ -\frac{2N}{R_0^2} B_{K_4} & \frac{N}{R_0} B_{K_4} & 0 & 0 & A_{K_{41}} & A_{K_{42}} & A_{K_{43}} & A_{K_{44}} \end{bmatrix} \quad (5.27)$$

where $A_{K_{ij}}$ is the (*i,j*) entry of the matrix $\boldsymbol{A_K}$; B_{K_i} is the i^{th} element of column vector $\boldsymbol{B_K}$; and C_{K_i} is the i^{th} element of row vector $\boldsymbol{C_K}$, all in eqn. (4.12). According to Theorem 4.2, the controller is stable if the eigen-values of the above-mentioned matrix A_{cl} have negative real parts.

5.4.3.1 Example

Using the numerical parameter values in Section 5.4.1, the weight functions for the RB-STU controller (see eqn. (5.23)) in the state-space representation (see eqn. (5.26)) can be determined as follows:

$$W_s(s) = \left[\frac{-\omega_b A}{\omega_b} \middle| \frac{1}{1/M}\right] = \left[\frac{4.5 \times 10^{-3}}{4.5} \middle| \frac{1}{0.2}\right]$$

$$W_T(s) = \left[\begin{array}{c|c} -1/a_3 & 1 \\ \hline a_2/a_3 & a_1/a_3 \end{array}\right] = \left[\begin{array}{c|c} \frac{-10}{30} & \frac{1}{0.01} \end{array}\right] \quad (5.28)$$

$$W_u(s) = \left[\begin{array}{c|c} 0 & 0 \\ \hline 0 & W_u \end{array}\right] = \left[\begin{array}{c|c} \frac{0}{0} & \frac{0}{10} \end{array}\right]$$

In addition, the designed RB-STU controller can be rewritten in state-space as follows:

$$K(s) = \left[\begin{array}{c|c} \boldsymbol{A_K} & \boldsymbol{B_K} \\ \hline \boldsymbol{C_K} & \boldsymbol{D_K} \end{array}\right] = \left[\begin{array}{cccc|c} -0.001111 & 0.0145 & 0.01147 & -0.2889 & 73.0261448 \\ 1.02235 & 3.3282 & 4.01598 & -117.31489 & 84.4991 \\ 0.498084 & 3.93726 & -15.95065 & 149.43775 & 34.70658 \\ 2.271587 & 13.04344 & 0.56860 & -53.26442 & 229.882 \\ \hline -0.00002746 & -0.0001256 & -0.00008029 & 0.00205883 & 0 \end{array}\right]$$

By substituting eqns. (5.28) and (5.29) in eqn. (5.27), one would obtain

$$A_{cl} = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -10.548 & -35.9555 & -12.3296 & 0 & 0 & -0.0000275 & -0.0001256 & -0.0000803 & 0.0020588 \\ -7995976.14 & 999497.0184 & 0 & -0.0045 & 0 & 0 & 0 & 0 & 0 \\ 6529394.5988 & -816174.324 & 0 & 0 & -10 & 0 & 0 & 0 & 0 \\ -275294731.8 & 34411841.48 & 0 & 0 & 0 & -0.001111 & 0.0145 & 0.01147 & -0.2889 \\ -318545734.57 & 39818216.82 & 0 & 0 & 0 & 1.02235 & 3.3282 & 4.01598 & -117.31489 \\ -130837267.98 & 16354658.49 & 0 & 0 & 0 & 0.498084 & 3.93726 & -15.95065 & 149.43775 \\ -866613072.74 & 1.08326634.09 & 0 & 0 & 0 & 2.271587 & 13.04344 & 0.56860 & -53.26442 \end{bmatrix}$$

from which the eigen-values are obtained as

$\lambda =$

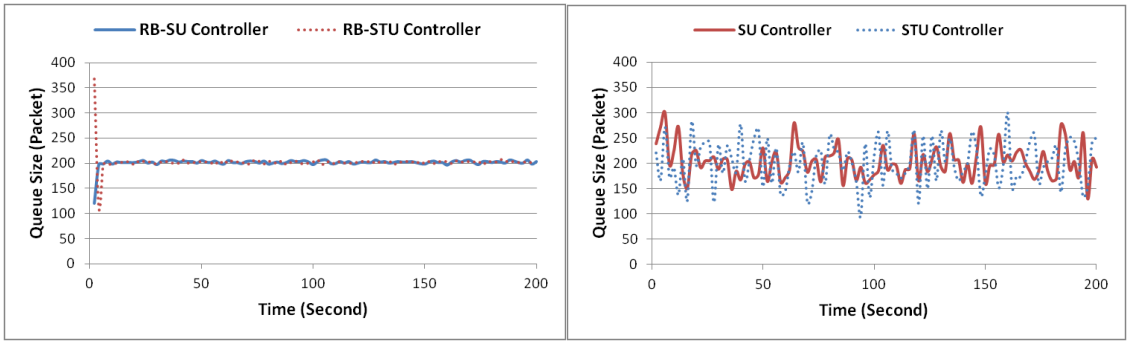
$[-0.0045 \quad -0.00739 \quad -8.936780 \pm 0.26425i \quad -9.169 \quad -9.99 \quad -32.877 \quad -51.191 \pm 5587088i]$

which are combinations of real and complex values but all with a negative real part. Therefore, $\boldsymbol{A_{cl}}$ is exponentially stable. According to Theorem 4.2, controller *K* would be called *stabilizing*.

5.5 Performance Evaluation

In this section, we shall use OPNET simulation [Opne13] to verify the capability of the proposed rate-based H_2/H_∞ controllers through a series of experiments. We use the same simulation settings and network topology from Section 4.5.1. Due to the difficulty in the RB-ST design commented earlier, we shall study different performances of the RB-SU and RB-STU controllers only. We have provided their comparison with the window-based SU and the STU controllers in Sections 5.5.1 to 5.5.4 in terms of their queue size, queueing delay, throughout and goodput, respectively. Note that we did not provide the comparison with the PI, SU H_∞ and STU H_∞ controllers because one can see the RB controllers are "much better" than the window-based counter parts and we already showed that the window-based controllers are better than the PI-controller and others. As clarified in Section 4.5.1, "Varying *N*" is understood to be "Varying

Network Load". Likewise, the N values in the figures of Section 5.5.7 are for the instantaneous number of connections.

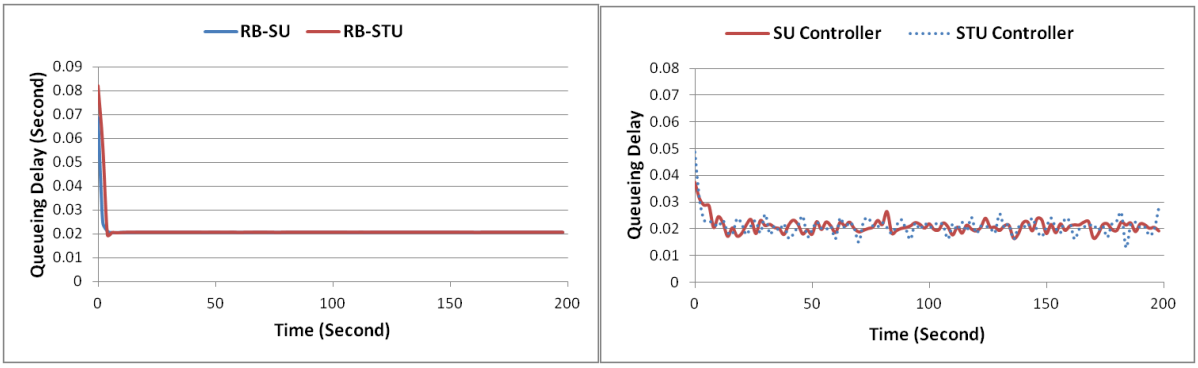


(a) **(b)**

Fig. 5.10: Instantaneous Queue Size Comparison of Different Controllers

5.5.1 Queue Size

Fig. 5.10 compares the instantaneous queue size among four different controllers. For more clarity, Fig 5.10a compares the RB-SU controllers with the RB-STU controllers while Fig 5.10b compares the SU controllers with the STU controllers. All controllers can reach their desire queue sizes. One can see the superiority of the rate-based controllers to the SU and STU controllers by inspecting the figures. Both the RB-SU controller and the RB-STU controller have relatively similar performance with hardly noticeable fluctuation (~σ_q in the steady-state is 4 and 3 for the RB-SU and the RB-STU respectively), and with short T_r (~2 sec and 4 sec for the RB-SU and the RB-STU respectively). The SU and STU controllers also have relatively small T_r (~ 6 sec and 2 sec respectively) but larger σ_q (~34 and 38 respectively).

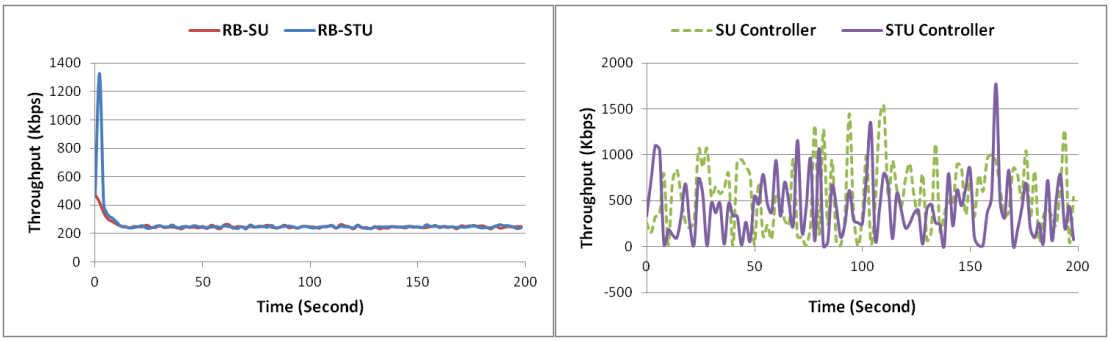


(a) **(b)**

Fig. 5.11: Queueing Delay Comparison of Different Controllers

5.5.2 Queueing Delay

Fig. 5.11 compares the queueing delay among four different controllers. All controllers attain the same queueing delay of 0.02 second in their steady states. However, it is apparent from the figure that there are significant differences in the steady state of the designed RB-SU and RB-STU controller from the SU and STU controllers. The designed rate-based controllers have very smooth queueing delay in the steady state where the SU and the STU controllers show fluctuations around the equilibrium point. Moreover, all controllers have small rise time/fall-times T_r although they are shorter for the rate-based controllers (~ 3 sec and 2 sec for the RB-SU and RB-STU controller respectively) when compared to the SU and STU controllers (~ 8sec and 9sec respectively).



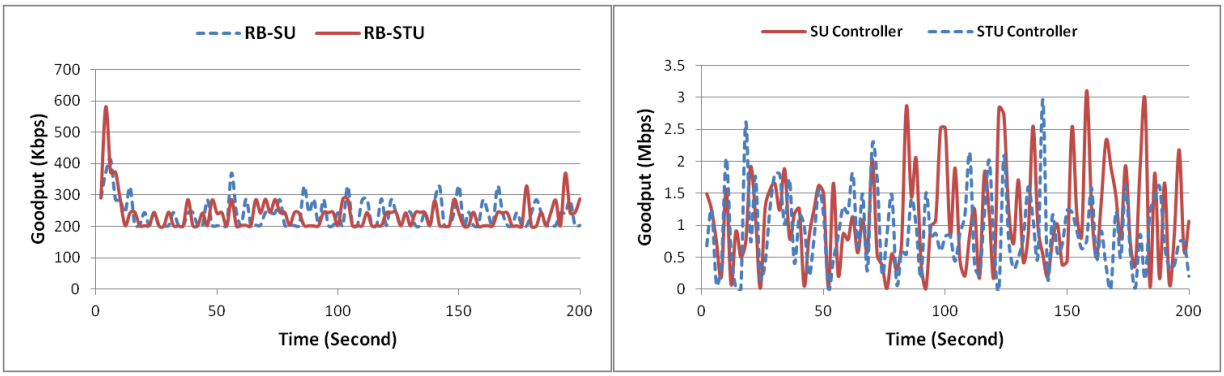
(a) **(b)**

Fig. 5.12: Throughput Comparison of Different Controllers

5.5.3 Throughput

Fig. 5.12 shows an example of the throughput (source sending rate) of one of the ftp sources (e.g. ftp 11) for the proposed rate-based controllers of RB-SU and RB-STU (Fig. 5.12a) and the window-based controllers of SU and STU (Fig. 5.12b). The SU controller and STU controller appear to have higher throughput (~ 548 Kbps and 414Kbps for SU and STU controllers respectively) than the RB-SU and RB-STU controllers (~263Kbps and 274Kbps for RB-SU and RB-STU controller respectively). However, both rate-based controllers have very smooth source sending rates and hence can support applications requiring a constant sending rate (e.g. streaming traffic). The throughputs of the window-based SU and STU controllers are not

smooth due to the changing congestion window size and hence the sending rate once a packet is dropped due to congestion.



(a) **(b)**

Fig. 5.13: Goodput Comparison of Different Controllers

5.5.4 Goodput

Fig. 5.13 shows the good-put comparison of different controller for an example source/destination pair (ftp 11). Recall from Section 4.5.1, goodput measures useful data at the destinations, without considering retransmitted data or the overhead of IP and TCP header from the source. Similar to source sending data rate shown in Fig. 5.12, the goodput for the rate-based controllers of RB-SU and the RB-STU are smoother than the window-based counter-parts of the SU controller and the STU controller. Again, the window-based controllers appear to have higher goodput on the average than the rate-based controllers (~1.12 Mbps for the SU controller, 0.95Mbps for the STU controller as opposed to 235 Kbps for the RB-SU controller and 247 for the RB-STU controller).

5.5.5 Sudden Bandwidth Changes

Since a deterministic amount of bandwidth is usually not available in contention-based network such as the wireless network, the bottleneck link can change suddenly. Therefore, we would like to study and compare the robustness capability of the designed rate-based controller with others using sudden bandwidth changes scenario of Fig 4.20.

Fig. 5.14 below compares the queue size of different controllers under bandwidth changes.

The RB-SU controller and RB-STU controller are able to maintain the queue size to the desired level of ~200 packets without noticeable changes in the queue size at the time of bandwidth changes. This can be attributed to the short rise/fall time of these controllers. Moreover, the queue size has very small fluctuations even in the presence of sudden bandwidth changes in the bottleneck link. Therefore, using rate-based H_2/H_∞ controllers for the system leads to a robust system under bottleneck bandwidth variation.

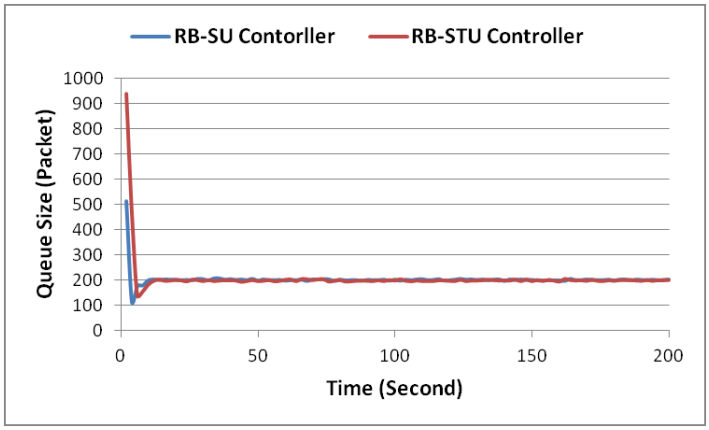


Fig. 5.14: Instantaneous Queue Size Comparison of Different Controllers Under Bandwidth Changes

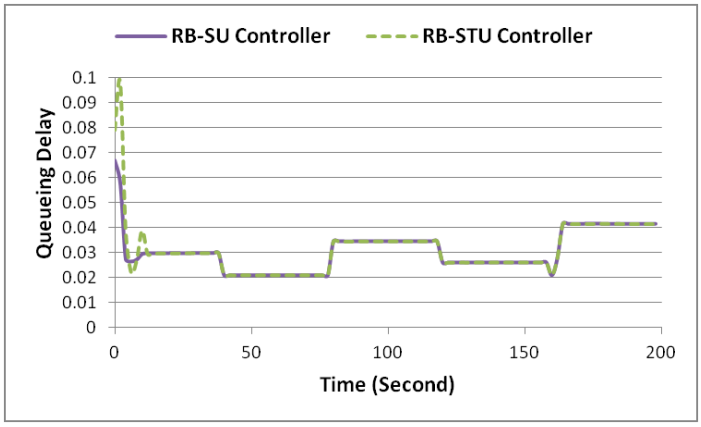


Fig. 5.15: Queueing Delay for the RB-SU and RB-STU Controllers Under Bandwidth Changes.

Fig. 5.15 compares the queueing delay of RB-SU and RB-STU controllers under the bandwidth changes. Since the both controllers have very small rise/fall times, they can reach their steady states before the link bandwidth changes. Note that the queueing delay follows the step changes. This is because the bandwidth is the router service rate, and hence the queueing

delay of the router is changed. For example, the highest bandwidth between t=40s to 80s gives the smallest queueing delay. Therefore, the proposed controllers have robust performance in the presence of bandwidth changes in the system.

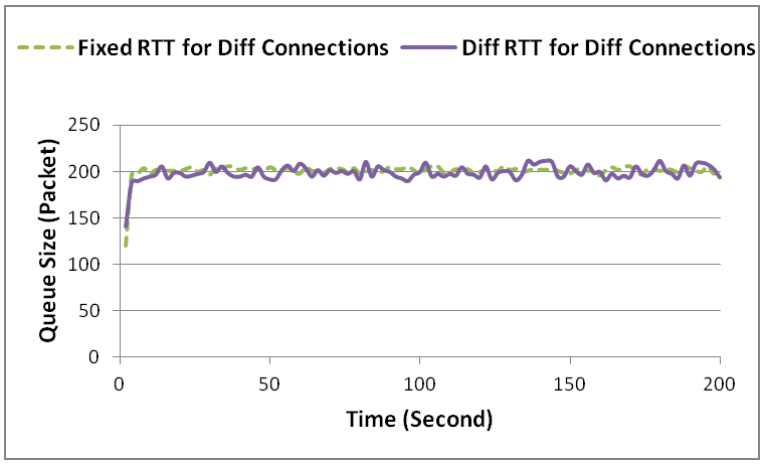


Fig. 5.16: Queue Size of RB-SU Controller Under Different RTT for Different Connections.

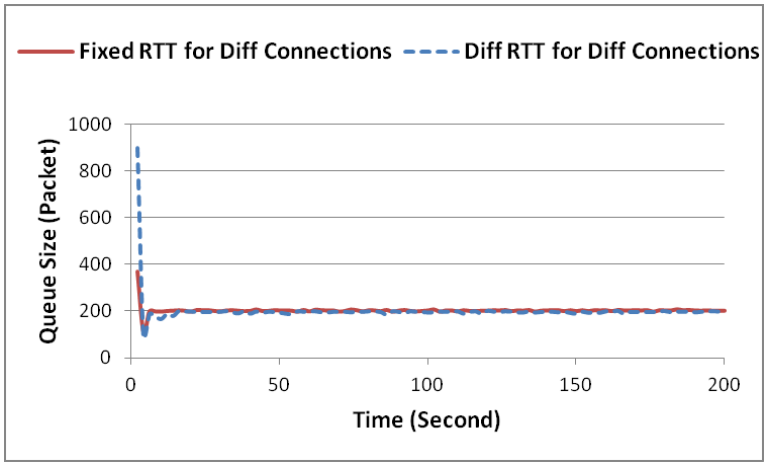


Fig. 5.17: Queue Size of RB-STU Controller Under Different RTT for Different Connections

5.5.6 Different RTTs for Different Connections

Since connections may not have the same RTT, we would like to investigate the robustness of the designed rate-based controllers under having different RTT for different connection by changing the RTPD for each link according to the scenario given in Section 4.5.2.7.

Fig. 5.16 compares the queue size of the RB-SU controller when the RTTs are different for different connections. It can be seen that the proposed RB-SU controller has a robust

performance because it is able to reach the desired queue size even when RTT is different for different connections. The tradeoff is that the queue size fluctuates more than the fixed RTTs scenario (~ σ_q is 8 for different RTTs scenario to 4 for fixed RTTs scenario).

Fig. 5.17 makes the same comparison for the RB-STU controller. The RB-STU controller works reasonably well under different RTTs; it is able to reach to the desired queue size with small rise/fall times (~3 sec) and has very small oscillations around the desire queue size. However, the initial queue size is higher (~900 packets) than that (~350 packets) in the different RTTs scenario.

5.5.7 Network Load Other Than The Nominal Value

Since a source may not be active (sending data to the network) all the time, we would like to investigate the robustness of the designed rate-based H_2/H_∞ controllers when the number of TCP connections is varying w.r.t. the nominal/designed value of N=100. We shall use the same source characteristic as discussed in Table 4.15 where there are 80 ftp sources sending data in the time interval of (0s, 40s), 100 in the interval of (40s, 80s), 120 in the interval of (80s, 120s), 70 in the time interval of (120s, 160s) and 60 connections in the time interval of (160s, 200s).

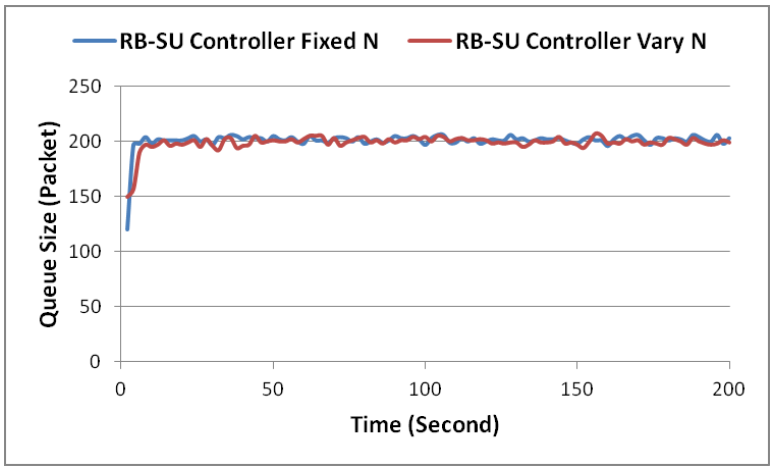


Fig. 5.18: Queue Size of RB-SU Controller Under the Varying Network Load Scenario

Fig. 5.18 compares the queue size of the RB-SU controller when the number of connections is fixed as opposed to when it is varying. The RB-SU controller works reasonably well; it is able to reach the desired queue size within small rise/fall times (~6 sec/4 sec). The queue size

oscillations are very small around the desired queue size level in both scenarios during the transitions to different network loads.

Fig. 5.19 makes the same queue size comparison for the designed RB-STU controller. The RB-STU controller performs reasonably well and robust; it is able to reach to desired queue size within small rise/fall time (~3sec). The queue size has very small oscillations around the desire queue size for both scenarios and unnoticeable changed during the transition time of different network loads. However, in this scenario the queue size starts from the higher value than the Fixed *N* scenario (~950 to 380).

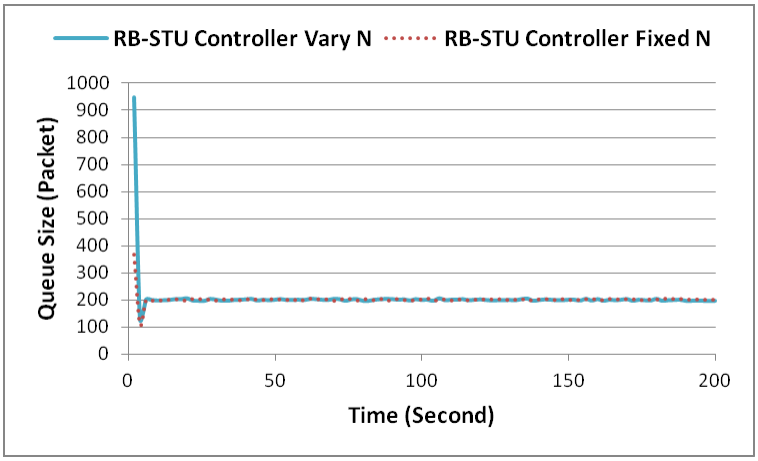
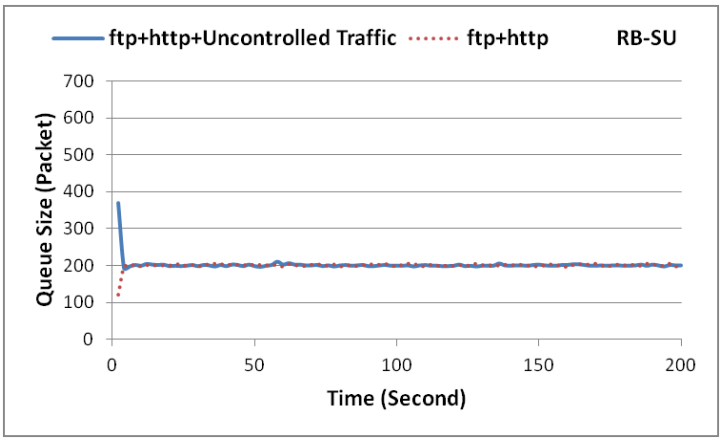


Fig. 5.19: Queue Size of RB-STU Controller Under the Varying Network Load Scenario

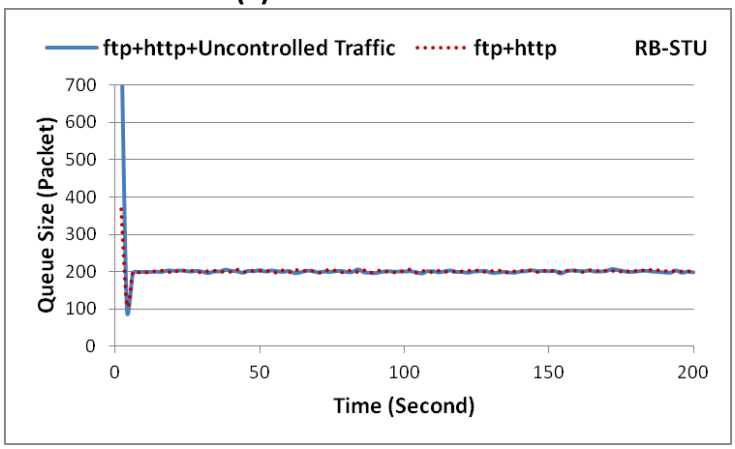
5.5.8 Disturbance from Uncontrolled Traffic

In order to investigate the behavior of our rate-based designed controllers user uncontrolled traffic, we have 20 uncontrolled ftp sources (unresponsive UDP-like) flowing into the network along with 80 ftp sources and 20 http sources, as done Section 4.5.2.8.

Fig. 5.20 below shows the queue size of the rate based controllers for different source's flows. It can be seen from the figure that the rate-based controllers perform reasonably well in the presence of uncontrolled traffic in the network; they are able to reach to the desired queue size in a very short rise/fall time (~4 sec for the RB-SU controller and 3 sec for the RB-STU controller). Therefore, the system can guarantee assigning a portion of bandwidth to the uncontrolled traffic without having effect on the system performance.



(a) RB-SU Controller



(b) RB-STU Controller

Fig. 5.20: Queue Size of RB-SU and RB-STU Controllers Under uncontrolled traffic scenario

5.6 Concluding Remarks

This chapter has designed and studied the rate-based H_2/H_∞ controller to control congestion in the TCP/AQM networks. Similar to the window-based H_2/H_∞ controllers, we have investigated different combinations of weight functions to design the RB-SU and RB-STU controllers. Both controllers have similar performance. We have not found an RB-ST controller with satisfactory performance. Like the window-based controllers, the ST controllers are difficult to design as their performances are mostly unstable. Their recommendation on the weight functions are summarized in Chapter 7.

Having stable queue and low delay are all fundamental ingredients for QoS design. We have demonstrated our rate-based Controllers can effectively enhance the performance of the

network of the window-based Controllers in Chapter 4. The rate-based controllers have significantly less fluctuations in queue size and queueing delay in the congested router. They have smoother sending rate (throughput) and goodput, although they have lower sending rate and goodput on average. Moreover, the rate-based controllers have shown robust performance under bandwidth changes scenario, having uncontrolled traffic, different RTT, different *N* scenarios. Since it was shown in Chapter 4 that the window-based Controllers have better performance than the PI, the SU H_∞ and the STU H_∞ controllers, the rate-based controllers naturally have better performance than these three controllers.

Chapter 6
Real-Time H_2/H_∞ Controller

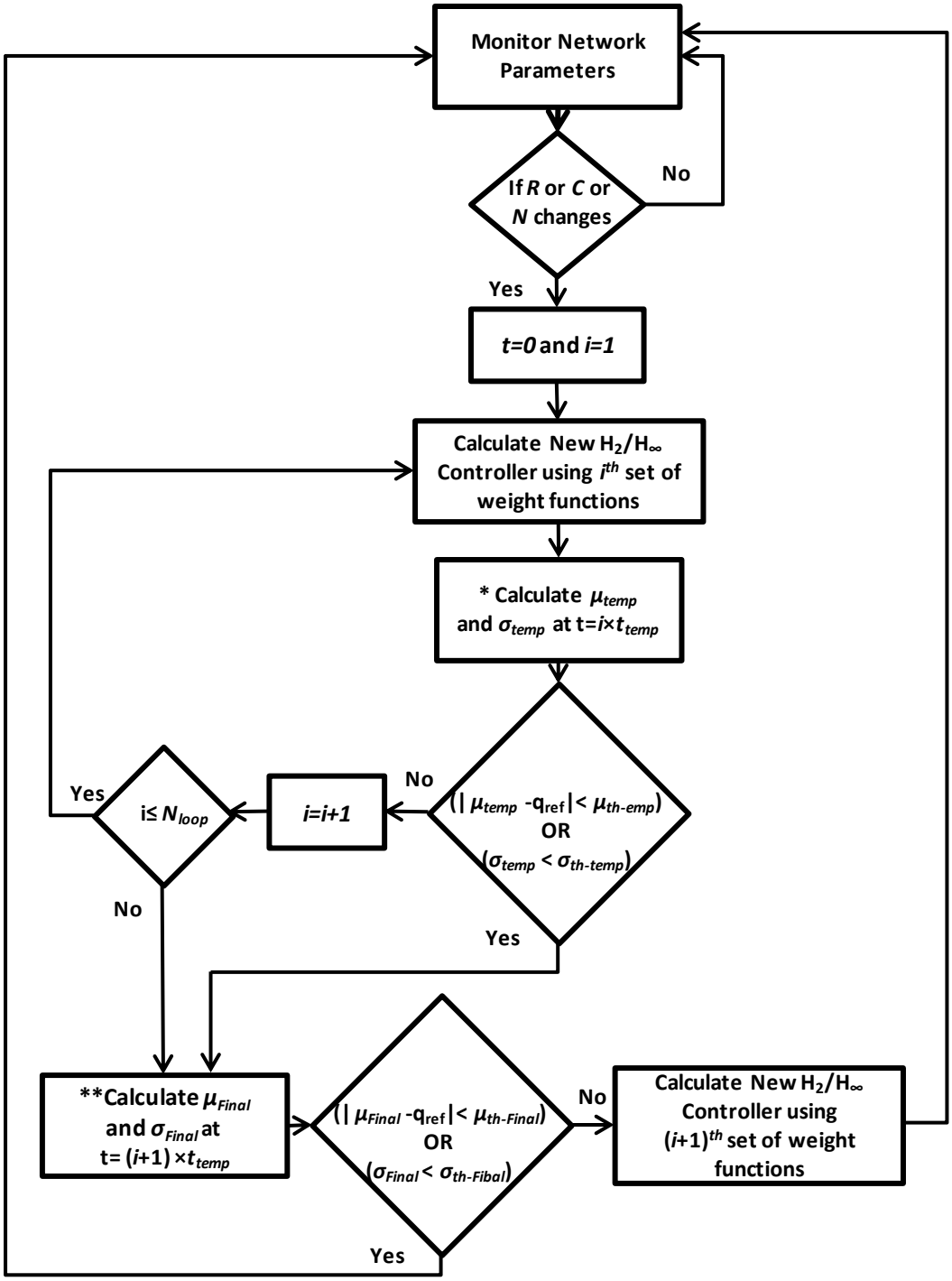
Performance evaluations (such as the rise/fall times) of the window-based controllers (Chapter 4) and rate-base controllers (Chapter 5) have demonstrated that these controllers can respond to the changes quite fast. It would be good if these features can be incorporated in a real time controller that can adapt to network parameters changes (like bandwidth and number of TCP sessions) with the hope to provide further performance improvement. Therefore, in this chapter, after we design the real time algorithms for different H_2/H_∞ controllers, we shall evaluate their performance and compare them with those in previous chapters under varying network parameters.

6.1 Real-Time Algorithm Design for the H_2/H_∞ Controllers

As discussed in the "static" design of previous chapters, we use the MATLAB [Matl13] to design the H_2/H_∞ controllers. This is captured in the procedure of Fig. 3.1 where one has to repeat the same procedure manually until an optimum design is obtained for the given set of network parameters. If any network parameter is changed, the procedure has to be repeated. The most time-consuming part is the manual entry of the controller parameters obtained from MATLAB into the OPNET controller algorithm before running the simulations to measure the performance.

We have now automated the above procedure in our real-time H_2/H_∞ Controller by calling the MATLAB [Matl13] design in the OPNET simulation. The OPENT simulation can call the MATLAB [Matl13] as an external function which will compute the coefficients in the numerator and denominator of the controller function *K(s)* right away for use in the OPNET controller algorithm and simulations. In other words, our real-time algorithm can produce a new controller that can match any changes in network parameters.

Before we describe/discuss the design algorithm for our real-time H_2/H_∞ controller in Fig. 6.1, we shall define/review a few parameters. We have R_0, C and N which are the RTT, capacity (bandwidth) and the number of TCP sessions of the network, respectively from before. Let μ_{temp}



*) μ_{temp} and σ_{temp} are the mean queue length and the standard deviation of the queue length calculated in time interval of t_{temp} respectively.

**) μ_{Final} and σ_{Final} are the mean queue length and the standard deviation of the queue length calculated from t=0 to t=(i+1)× t_{temp} respectively.

Fig. 6.1: Design Algorithm for the Real-Time H_2/H_∞ Controller

be the mean queue length and σ_{temp} be the standard deviation of the queue length in a time interval of t_{temp} seconds. Moreover, let $\mu_{th\text{-}temp}$ and $\sigma_{th\text{-}temp}$ be the temporary threshold values for the mean and the standard deviation of the queue length respectively with q_{ref} as the reference/desired queue value (such as those used in previous chapters). Let N_{loop} be the maximum number of times the design of a controller is iterated if the loop condition ($|\ \mu_{temp} - q_{ref}\ | < \mu_{th\text{-}temp}$) or ($\sigma_{temp} < \sigma_{th\text{-}temp}$) is not met. Also, let μ_{Final} and σ_{Final} be the mean queue length and the standard deviation of the queue length from t=0 to t=(i+1)× t_{temp} seconds after the network parameter has been changed where i is the number of iterations.

As depicted at the top of Fig. 6.1, the procedure is self-looping forever if there are no changes in the monitored network parameters. Whenever anyone of the network parameters is changed, a new controller design loop is initiated. After a duration of t_{temp} seconds, the mean length μ_{temp}, of the queue and its standard deviation σ_{temp} during this duration are calculated to see if they are within the desired range of the desired queue length q_{ref} (as indicated by the big diamond box in the middle of the figure).

If any conditions of ($|\ \mu_{temp} - q_{ref}\ | < \mu_{th\text{-}temp}$) or ($\sigma_{temp} < \sigma_{th\text{-}temp}$) is not satisfied, the next set of parameter values will be selected to design a new controller *K(s)* to control traffic for the next t_{temp} seconds. Then the same checking process is applied by evaluating μ_{temp} and σ_{temp}. This process will be repeated up to a maximum of N_{loop} times. If the numbers of iterations i exceeds N_{loop} or the above condition is satisfied, a similar but final checking process is applied by evaluating μ_{Final} and σ_{Final} to see if they are within an acceptable ranges of $\mu_{th\text{-}Final}$ and $\sigma_{th\text{-}Final}$ (the big diamond box in the bottom of Fig. 6.1) at t=(i+1)× t_{temp} seconds.

If the final condition check is not satisfied, a new controller simply uses the next set (the $(i+1)^{th}$ set) of the weight functions for design. The next set is used for simplicity reason because the results of previous design cases were not saved. After all, our experience is that the algorithm hardly exhausts all design cases to find a suitable one. Note that during the execution time of this algorithm the network parameters are continuously monitored. If changes occur, the algorithm is started all over from the beginning.

Using the parameter values of the weight functions (W_s, W_U and/or W_T) of the SU, ST and STU controllers determined in Chapters 4 and 5, our RT (Real-Time) controller can be preloaded

with selected sets of these values that would likely give good performance for the implemented controllers

Note that the values of t_{temp}, $\mu_{th\text{-}temp}$, $\sigma_{th\text{-}temp}$, $\mu_{th\text{-}Final}$, $\sigma_{th\text{-}Final}$ and N_{loop} are design/pre-determined values chosen by the designer. The mean queue length μ_q measured from the queueing performance for each controller in Chapters 4 and 5 can be used as a guideline for choosing the $\mu_{th\text{-}temp}$ value such that the μ_{temp} measured in each loop can be gauged by the condition $| \mu_{temp} - q_{ref} | < \mu_{th\text{-}temp}$. We also use the measured σ_q for each controller as a guideline to choose a value for the bound, $\sigma_{th\text{-}temp}$ such that $\sigma_{temp} < \sigma_{th\text{-}temp}$ for the measured σ_{temp} in each loop. Note that we set a narrower range for the final evaluating condition, i.e. $\mu_{th\text{-}Final} < \mu_{th\text{-}temp}$ and $\sigma_{th\text{-}Final} < \sigma_{th\text{-}temp}$, to see if the last designed controller is able to show a better performance. If so, this controller is used until the new network parameters changes is monitored

6.2 Real-Time Window-Based H_2/H_∞ Controller

The procedure of Fig. 6.1 is now used to design the real-time window-based H_2/H_∞ controllers for the TCP/AQM network. We need to choose sets of parameter values to implement the weight functions for the plant model in Section 4.1 from which a specific controller is designed. Take the real-time SU controller for example. We need to identify the workable values of W_u and W_s (which includes three parameters M, A and ω_B) from Chapter 4 and propose (N_{loop}+1) sets of (W_u, M, A, ω_B) from the combinations of these workable values based on our results and experience. These selected sets are chosen the recommendation range for SU controller in Chapter 7. For the SU controller we always have N_{loop}+1 = 6 sets of (M, ω_B, A, W_u) selected in the following order (2, 5.5, 10^{-7}, 10^5), (1.5, 6, 10^{-6}, 8×10^4), (2, 6.5, 10^{-5}, 2×10^5), (0.7, 7, 10^{-7}, 9×10^4), (1, 5.5, 10^{-6} ,10^5), (0.7, 6,10^{-5} , 10^4). The first set is always the best set identified in Section 4.2.2 while the last set is used when no condition is satisfied after N_{loop} iterations. Note that we have also tried other sets from the recommended parameter values in Chapter 7 and did not find any significant differences.

As mentioned in Section 6.1, the values of t_{temp}, $\mu_{th\text{-}temp}$, $\sigma_{th\text{-}temp}$, $\mu_{th\text{-}Final}$, $\sigma_{th\text{-}Final}$ and N_{loop} are determined by the designer. We choose t_{temp}=1 sec which is frequent enough to check the performance of the controller. Moreover, we determine $\mu_{th\text{-}temp}$=30, $\sigma_{th\text{-}temp}$=10, $\mu_{th\text{-}Final}$=5 and

$\sigma_{th\text{-}Final}$=5. These values are chosen according to window-based controllers' performance in Chapter 4. For example, since the standard deviation of the designed controllers from Chapter 4 is about 30 packets, we have chosen a tighter boundary of $\sigma_{th\text{-}temp}$=10 in order to have better performance for the real-time controller. We have tried other values for $\mu_{th\text{-}temp}$, $\sigma_{th\text{-}temp}$, $\mu_{th\text{-}Final}$, $\sigma_{th\text{-}Final}$ and the performance of the real-time controllers are not significantly sensitive to these values. Similarly, we have chosen the following for other controllers:

a) Real-time ST Controller: we have N_{loop}+1 = 6 sets of (M, ω_B,, A, α_1, α_2, α_3) in the following order (2, 5.5, 10^{-7}, 0.001, 3, 0.1), (1.5, 5, 10^{-6}, 0.01, 3.5, 0.1), (0.7, 6 , 10^{-4}, 0.05, 4, 0.3), (1, 6.5, 10^{-5}, 0.001, 3, 0.5), (1.5, 7, 10^{-6}, 0.005, 2, 0.1), (2, 5, 10^{-7}, 0.1, 3.5, 0.3) from Section 4.3.2. We also choose t_{temp}=1 sec, $\mu_{th\text{-}temp}$=30, $\sigma_{th\text{-}temp}$=15, $\mu_{th\text{-}Final}$=5 and $\sigma_{th\text{-}Final}$=7 based on our performance results in Section 4.3.2.1.

b) Real-time STU Controller: we have N_{loop}+1 = 6 sets of (W_u, ω_B, M, A, α_1, α_2, α_3) in the following order (10^5, 2, 5.5, 10^{-7}, 0.001, 6, 0.01), (8×10^4, 1.5, 6, 10^{-6}, 0.01, 2, 0.001), (2×10^5, 2, 6.5, 10^{-5}, 1, 2.5, 0.01), (9×10^4, 0.7, 7, 10^{-7}, 0.001, 3.5, 0.001), (10^5, 1, 5.5, 10^{-6}, 0.01, 6, 0.01), (10^4, 0.7, 6, 10^{-5}, 1, 7, 0.001) from Appendix F. We also choose t_{temp}=1 sec, $\mu_{th\text{-}temp}$=30, $\sigma_{th\text{-}temp}$=15, $\mu_{th\text{-}Final}$=5 and $\sigma_{th\text{-}Final}$=7 based on our performance results in Appendix F.1.

Again, we have also tried other sets from the recommended parameter values in Chapter 7 and did not find any significant differences.

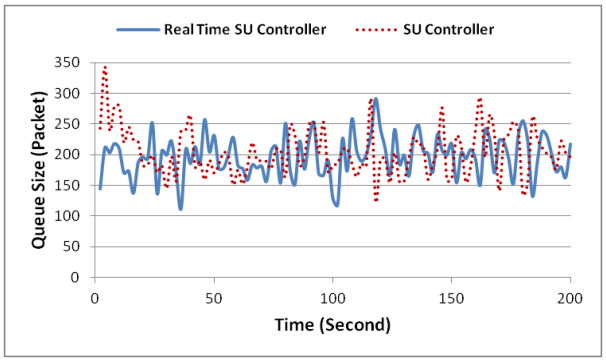
6.2.1 Performance Evaluation

After equipping our OPNET simulator with the capability to call MATLAB functions real-time for the redesign of different parameter values to achieve optimization (of the objective function in Section 3.1), we can now conduct simulation experiments to test the capability of the real-time window-based controllers in responding to network changes. We are hoping to see improvements over the previous "static" controller designs.

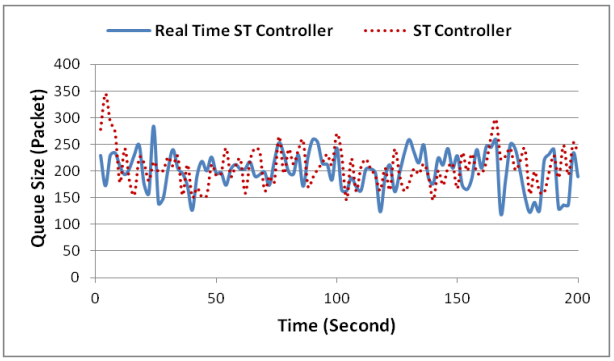
In order to make comparison, we shall use the same network topology in Fig. 4.12, and the same network parameters mentioned in Section 4.5 as the non-real-time controllers. Our experiments are conducted using OPNET Modeler 17.5 [Opne13] on an Intel Core i5 platform with a 1 GHz CPU. The simulation time would depend on the bottleneck bandwidth and the

simulated time. For a typical simulated time of 200 seconds in our experiments, the simulation time usually takes about 16 minutes which an order of 1 million packets is collected for our statistics. For every time the simulator calls MATLAB to find the new controller, about 500ms is needed that the new controller coefficients launched in to the simulator; this is shorter than the rise/fall time measured in Chapters 4 and 5.

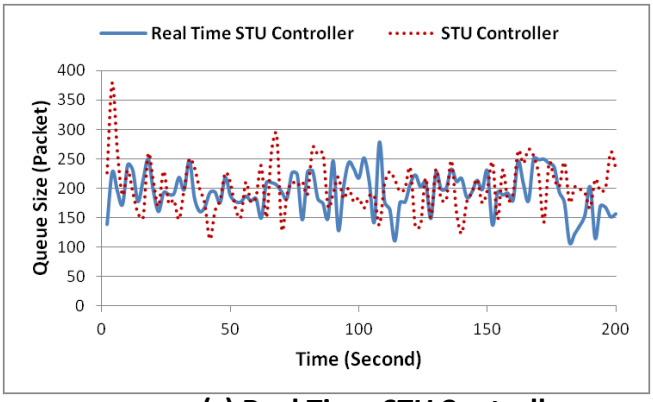
Since the real time controller is proposed to work under varying network environment, we shall verify the performance when the number of instantaneous TCP connections, or when the bandwidth *C* is changing suddenly, and compare it with the regular window-based controller. Finally, we shall evaluate and compare their performances when both *N* and *C* are changing.



(a) Real Time *SU* Controller



(b) Real Time *ST*



(c) Real Time *STU* Controller

Fig. 6.2: Queue Size of the Real-Time SU, ST and STU Controllers under Sudden Bandwidth Changes and the Comparison to Their Non-Real-Time Counterparts

6.2.1.1 Sudden Bandwidth Changes

In order to investigate the performance of the proposed real-time controller, we shall evaluate its capability to sudden bandwidth changes like the scenario given in Section 4.5.2.6 in Fig. 4.20.

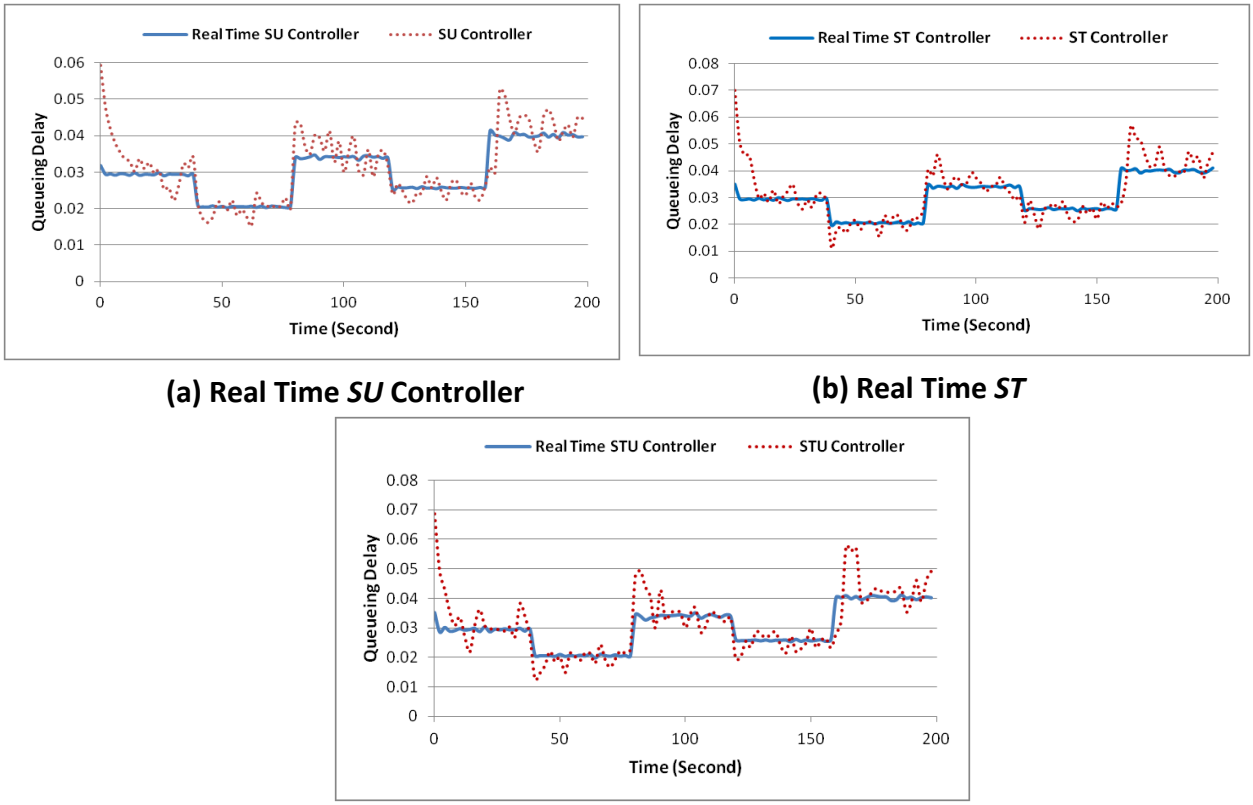
We compare queue size and queueing delay of the real-time STU controller to STU, real-time ST controller to the ST and real-time SU controller to the SU controller.

a) Queue Size

Fig. 6.2 compares the queue size of different controllers under bandwidth changes. All the real-time window-based controllers work better than window-based ones. They are able to reach to desired queue size at smaller rise/fall time (~1 sec for the real-time SU controller compared to 19 sec for the SU controller and 1s for the real-time ST controller compared to 9sec for the ST controller and 1sec for the real-time STU controller to 8sec for the STU controller). They also have smaller peak queue values (~250packets for the real-time SU controller compared to 350 packets for the SU controller; 255 packets for the real-time ST controller compared to 350 packets for the ST controller; and 248 packets for the real-time STU controller to 380 packets for the STU controller) with smaller queue size oscillation by having a smaller σ_q (~28packets for the real-time SU controller compared to 39 packets for the SU controller; 32 packets for the real-time ST controller compared to 41 packets for the ST controller; and 29 packets for the real-time STU controller to 36 packets for the STU controller).

b) Queueing Delay

Fig. 6.3 below compares queueing delay of different controllers under the bandwidth changes. One can see that the real-time controllers have smoother queueing delay in each step and have smaller step changes fluctuation suggesting that the real-time controllers are more robust to bottleneck bandwidth changes. In fact, the delay values do not settle down well in the non-real-time controllers before the network condition change again while the real-time can. For example, it takes 1 sec for the real-time SU controller to queueing delay settles down when compared to 19 sec for the SU controller. Also, it takes 1s for the real-time ST controller compared to 9sec for the ST controller and 1sec for the real-time STU controller to 8sec for the STU controller. This is because the real-time controllers can redesign the controllers according to the parameters changes. One can also see that there is only a slight difference among the different kinds of real-time window-based controllers' reaction to step bandwidth changes because they all have very small rise/fall time.



(a) Real Time *SU* Controller

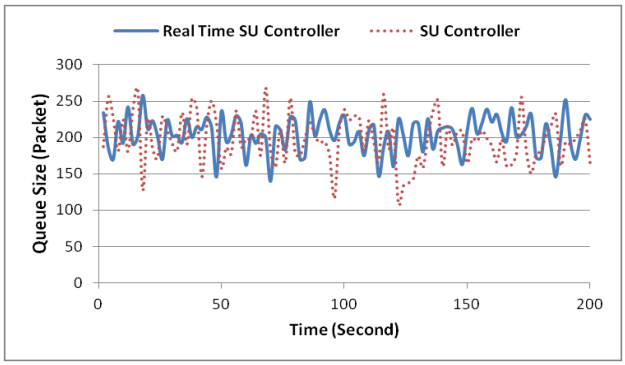
(b) Real Time *ST*

(c) Real Time *STU* Controller

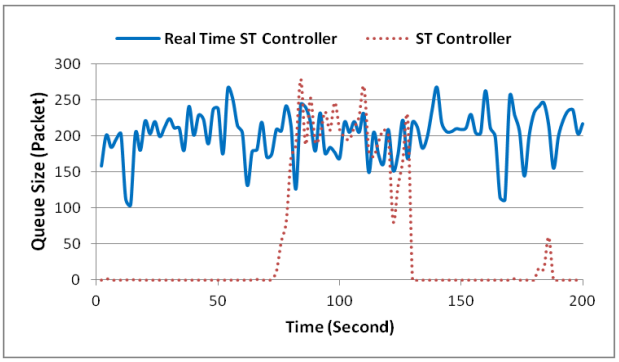
Fig. 6.3: Queueing Delay of the Real-Time SU, ST and STU Controllers under Sudden Bandwidth Changes and the Comparison to Their Non-Real-Time Counterparts

6.2.1.2 Changing Network Load Scenario

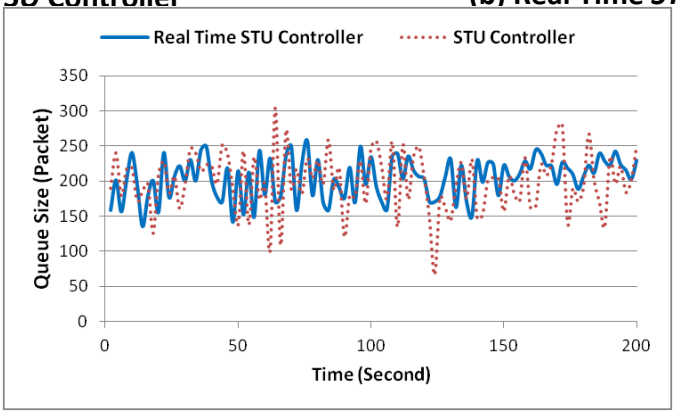
In this section, we shall investigate the performance of the real-time controllers when the network load is changing according to the description in Section 4.5.2.9.a. In another words, there are 80 ftp sources sending data in the time interval of (0s, 40s), 100 sources in the interval of (40s, 80s), 120 in the interval of (80s, 120s), 70 in the time interval of (120s, 160s) and 60 in the time interval of (160s, 200s).



(a) Real Time *SU* Controller



(b) Real Time *ST* Controller

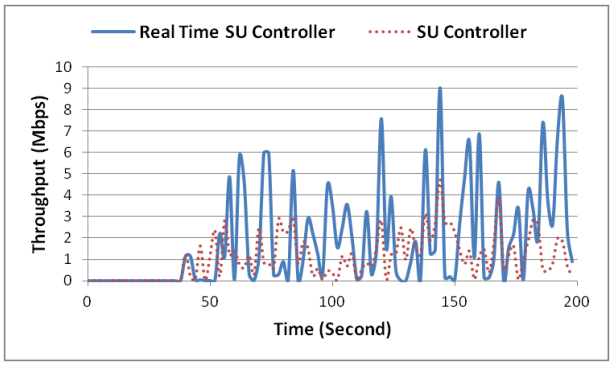


(c) Real Time *STU* Controller

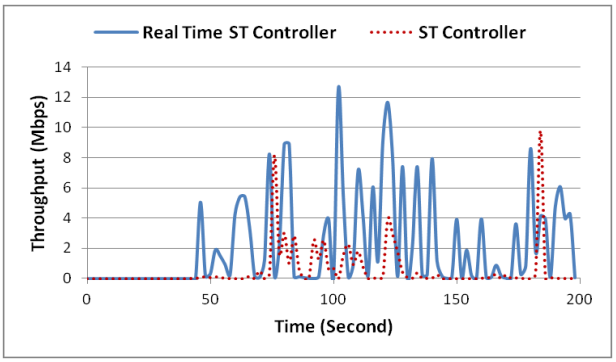
Fig. 6.4: Queue Size of the Real-Time SU, ST and STU Controllers under Varying Network Load and the Comparison to Their Non-Real-Time Counterparts

a) Queue Size

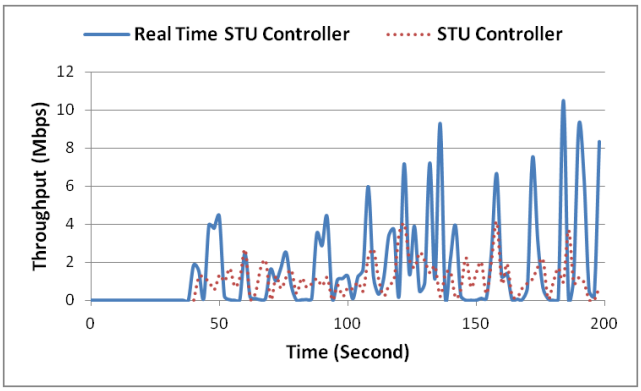
Fig. 6.4 compares the queue size of different controllers under changing *N* scenario. It can be seen from the figure that all the real-time window-based controllers work better than window-based ones. The real-time SU controller has smaller oscillation to the SU controller (~σ_q= 24packets for the real-time SU controller when compared to 33 packets for the SU controller). The real-time STU controller also has smaller oscillation compared to the STU controller (~σ_q=29packets for the real-time STU controller compared to 41 packets for the STU controller). One can see that, the ST controller was not able to reach to the desire queue value while real-time ST controller shows reasonable performance under Changing *N* scenario, by reaching to desired queue value all the time.



(a) Real Time *SU* Controller



(b) Real Time *ST* Controller



(c) Real Time *STU* Controller

Fig. 6.5: Throughput of the Real-Time SU, ST and STU Controllers under Varying Network Load and the Comparison to Their Non-Real-Time Counterparts

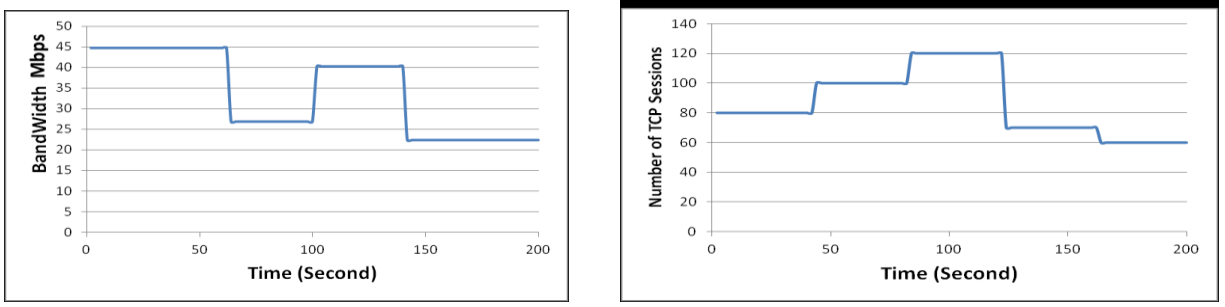
b) Throughput

Fig. 6.5 compares an example of the throughput (source sending rate) of one ftp sources (e.g. ftp 11) for different controllers under changing network load scenario. According to Table 4.15, ftp 11 starts sending date from t=40 sec. It can be seen from the figure that all the real-time window-based controllers have higher throughput than the window-based ones (~2.3 Mbps for the real-time SU controller when compared to 1.2 Mbps for the SU controller, 2.4 Mbps for the real-time ST controller compared to 0.9 Mbps for the ST controller and 2.1 Mbps for the real-time STU controller compared to 1 Mbps for the STU controller).

6.2.1.3 Varying Network Load and Sudden Bandwidth Changes Scenario

In this section, we shall investigate the performance of the real-time controllers when both the number of the TCP sessions *N* and the network bandwidth *C* are changing. We shall compare the queue size, queueing delay and the throughput between the real-time and non-real-time

controllers.



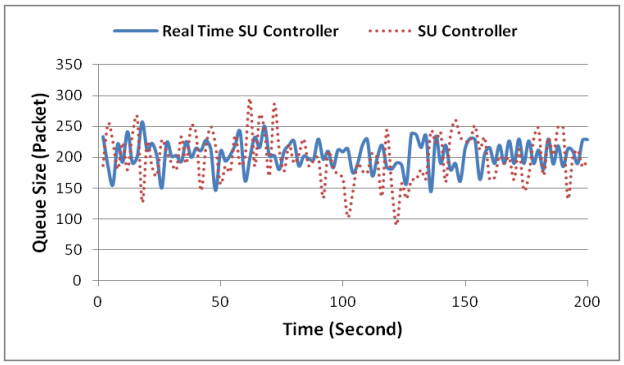
(a) Bandwidth **(b) Number of TCP Sessions,**

Fig 6.6: Bandwidth and Number of TCP Sessions Variation

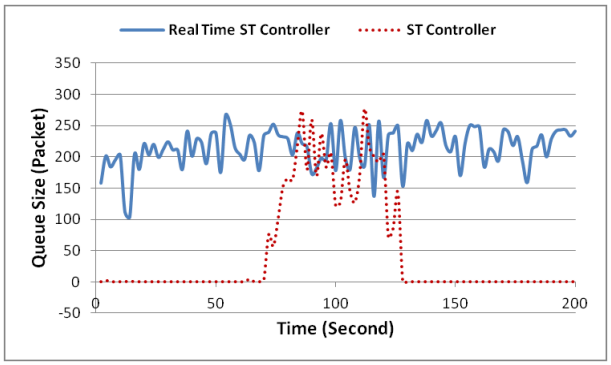
Fig. 6.6 shows a scenario where the bandwidth varies in steps between 22 and 45 Mbps. The biggest change occurs at t=60s and 140s when the bandwidth drops from 45 Mbps to 27 Mbps (a 40% change) at t=60s and from 40Mbps to 22Mbps at t=140s (a 44% change). Meanwhile, the number of TCP sessions is also changing in a manner similar to Section 6.2.2.1 and 4.5.2.9.a.

a) Queue Size

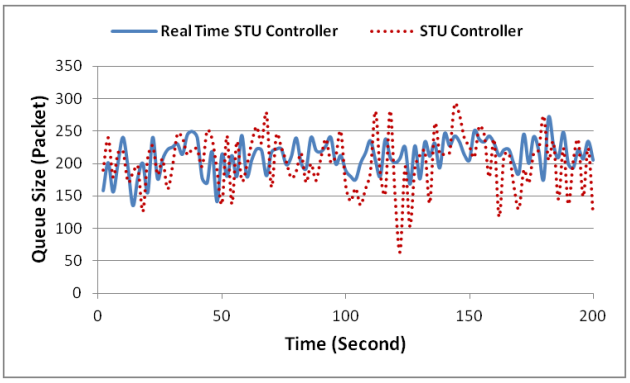
It can be seen from Fig. 6.7 below that unlike the non-real-time window-based controllers, all the real-time window-based controllers are able to show reasonable performance. The real-time SU controller has smaller oscillations (~σ_q= 23packets compared to 39 packets for the non-real-time SU controller). Same observations are made for the comparison of STU controllers (~σ_q=26 packets compared to 42 packets for the non-real-time STU controller). Quite noticeable is the real-time ST controller which now, has desirable queue value all the time with a small σ_q=31packets, while the non-real-time version was not able to reach to the desire queue value all the time.



(a) Real Time *SU* Controller



(b) Real Time *ST* Controller

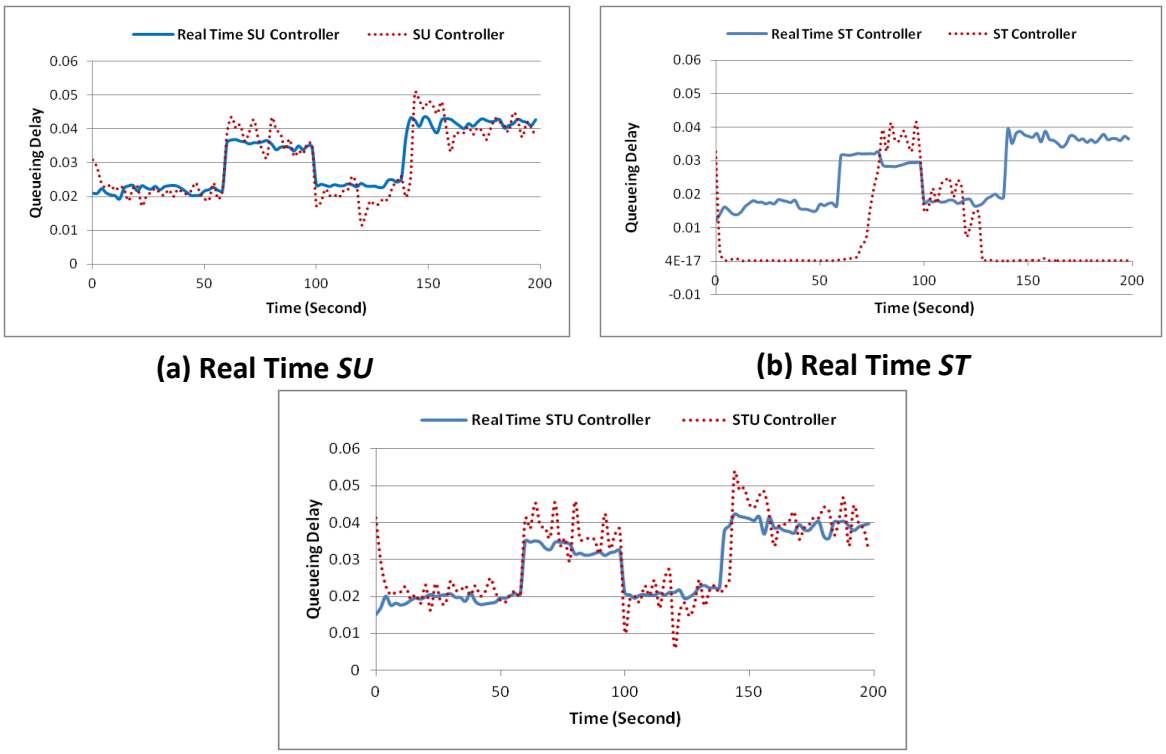


(c) Real Time *STU* Controller

Fig. 6.7: Queue Size of the Real-Time SU, ST and STU Controllers under Varying Network Load and Bandwidth Changes and the Comparison to Their Non-Real-Time Counterparts

b) Queueing Delay

Fig. 6.8 below compares the queueing delay of different controllers. The real-time controllers have smoother queueing delay in each step and have smaller overshoots due to step changes, thus suggesting the real-time controllers are more robust to bottleneck bandwidth changes. In fact, the non-real-time controllers cannot settle their queueing delay down before the network conditions change again. Like the queueing delay in Fig.6.7, the ST controller cannot follow the changes and is not robust.



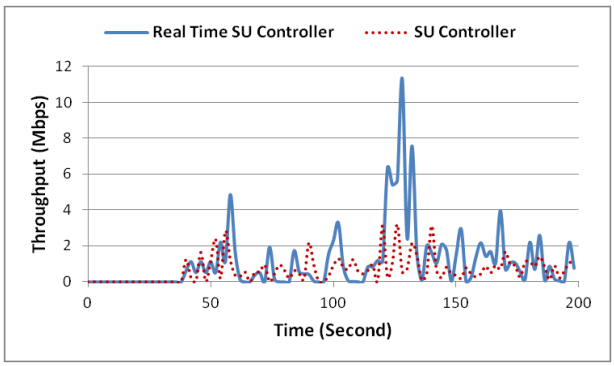
(a) Real Time *SU* **(b) Real Time *ST***

(c) Real Time *STU*

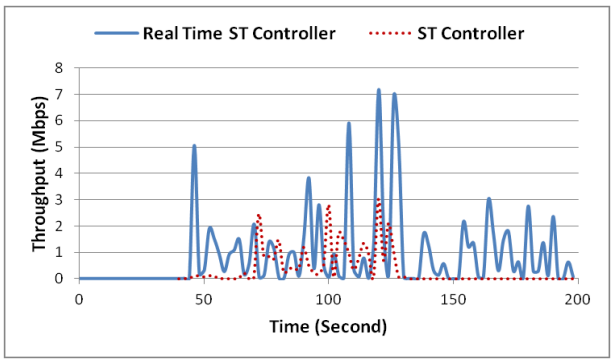
Fig. 6.8: Queueing Delay of the Real-Time SU, ST and STU Controllers under Varying Network Load and Bandwidth Changes and the Comparison to Their Non-Real-Time Counterparts

c) Throughput

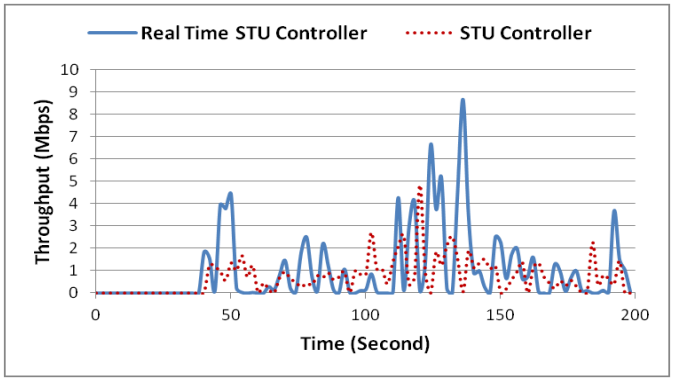
Fig. 6.9 compares an example of the throughput (source sending rate) of one ftp sources (e.g. ftp 11) for different controllers. According to Table 4.15, ftp-11 stream starts sending data from t=40 sec. It can be seen from the figure that all real-time window-based SU controllers have higher throughput (~2.1 Mbps compared to 1.1 Mbps for the non-real-time SU controller). Same observation is made for the real-time ST controllers (~2Mbps for the real-time ST controller compared to 0.7 Mbps of the non-real-time ST controller) and the real-time STU controllers (1.8 Mbps compared to 0.9 Mbps for the non-real-time STU controller).



(a) Real Time *SU* Controller



(b) Real Time *ST* Controller



(c) Real Time *STU* Controller

Fig. 6.9: Throughput of the Real-Time SU, ST and STU Controllers under Varying Network Load and Bandwidth Changes and the Comparison to Their Non-Real-Time Counterparts

6.3 Real-Time Rate-Based H_2/H_∞ Controller

Similar to the real-time window-based controller in Section 6.2, we also use the procedure of Fig. 6.1 to design the real-time version of the rate-based controllers for the TCP/AQM network in Chapter 5. We need to choose sets of parameter values to implement the weight functions for the plant model in Section 5.1 from which a specific controller is designed. The following are the sets and other parameters values we choose for our real-time controllers.

a) Real-time RB-SU Controller: we have $N_{loop}+1 = 6$ sets of (M, ω_B, A, W_u) in the following order (5, 4.5, 10^{-3}, 10), (5.1, 3, 10^{-2}, 20), (5.2, 5, 10^{-3}, 100), (5.3, 6, 10^{-3}, 10), (5.4, 4.5, 10^{-2}, 50), (5, 6, 10^{-3}, 70) from Section 5.3.2. We also choose t_{temp}=1 sec, $\mu_{th\text{-}temp}$=15, $\sigma_{th\text{-}temp}$=7, $\mu_{th\text{-}Final}$=5 and $\sigma_{th\text{-}Final}$=5 based on our performance results in Section 5.5.1.

b) Real-time RB-STU Controller: we have $N_{loop}+1 = 6$ sets of (W_u, M, ω_B, A, a_1, a_2, a_3) in the following order (10, 5, 4.5, 10^{-3}, 0.001, 3, 0.1), (20, 5.1, 3, 10^{-2}, 0.002, 2.5, 0.01), (100, 5.2, 5, 10^{-}

^{3}, 0.01, 1, 0.05), (10, 5.3, 6, 10^{-3}, 0.1, 2.75, 0.1), (50 , 5.4, 4.5, 10^{-2}, 1, 3.5, 0.01), (70, 5, 6, 10^{-3}, 0.001, 3, 0.05) from Appendix I. We also choose t_{temp}=1 sec, $\mu_{th\text{-}temp}$=15, $\sigma_{th\text{-}temp}$=7, $\mu_{th\text{-}Final}$=5 and $\sigma_{th\text{-}Final}$=5 based on our performance results in Appendix I.

Again, more sets from the selected values in Chapter 7 have been tried and there is no significant difference. Likewise, we have tried different values for t_{temp}, $\mu_{th\text{-}temp}$, $\sigma_{th\text{-}temp}$, $\mu_{th\text{-}Final}$, $\sigma_{th\text{-}Final}$ and N_{loop} and the evaluations show the performance is not significant sensitive.

Note that the nominal values for the network parameters are N=100 greedy ftp sources using TCP sessions along with 20 http sources, a round trip time R_0=0.25s, and the bandwidth of the bottleneck link at C=9708 packets/second. The real-time algorithm starts working when the network parameters changes.

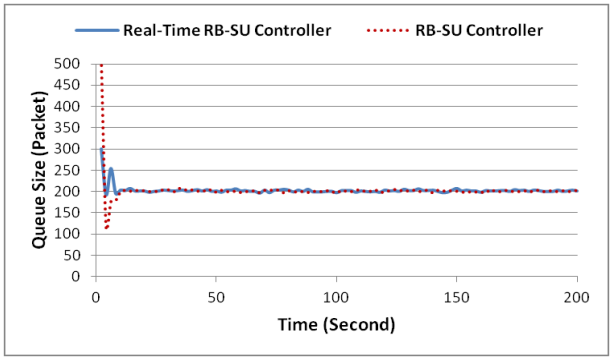
6.3.1 Performance Evaluation

In order to make comparison, we shall use the same network topology in Fig. 4.12. Similar to Real-Time Window-based controller mentioned in Section 6.2.1, we conduct our experiments using OPNET Modeler 17.5 [Opne13] on an Intel Core i5 platform with a 1 GHz CPU. For a typical simulated time of 200 seconds in our experiments, the simulation time usually takes about 1 hour and 17 minutes during which an order of 1 million packets is collected for our statistics. For every time the simulator calls MATLAB to find the new controller, about 500ms is needed that the new controller coefficients launched in to the simulator.

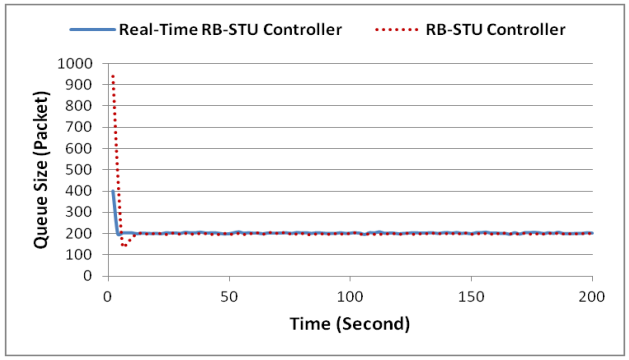
Similar to Section 6.2, the performance of the real-time rate-based controller shall be evaluated under three scenarios, Varying N, Varying C and a scenario when both C and N are changing. The comparison to RB-ST controller is omitted as we were not able to obtain stable results for such controller.

6.3.1.1 Sudden Bandwidth Changes

In order to investigate the performance of the proposed real-time controller, we shall evaluate its capability to sudden bandwidth changes like the scenario given in Section 4.5.2.6 in Fig. 4.20. We compare the queue size and queueing delay of the real-time RB-STU to the RB-STU controller, as well as the real-time RB-SU controller to the RB-SU controller.



(a) Real-Time RB-SU Controller

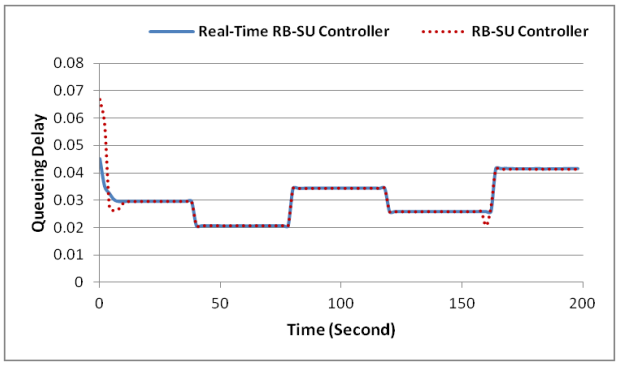


(b) Real-Time RB-STU

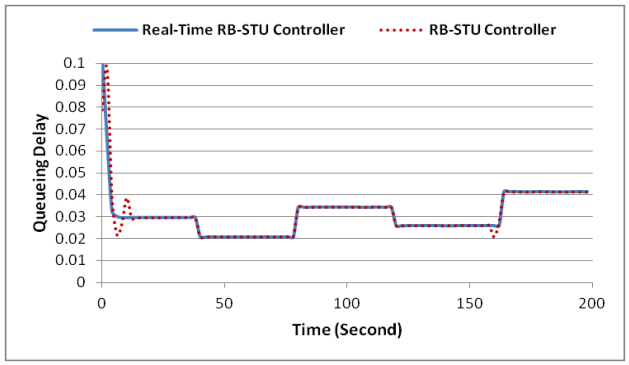
Fig. 6.10: Queue Size of the Real-Time RB-SU and RB-STU Controllers under Sudden Bandwidth Changes and the Comparison to Their Non-Real-Time Counterparts

a) Queue Size

Fig. 6.10 compares the queue size of different controllers under bandwidth changes. The real-time controllers show smaller peak values (~300packets for the real-time RB-SU controller when compared to 500 packets for the RB-SU controller; 400 packets for the real-time RB-STU controller to 920 packets for the RB-STU controller). At steady-state, all the controllers show good performance with non-significant oscillations while bottleneck bandwidth is changing.



(a) Real-Time RB-SU Controller



(b) Real-Time RB-STU

Fig. 6.11: Queueing Delay of the Real-Time RB-SU and RB-STU Controllers under Sudden Bandwidth Changes and the Comparison to Their Non-Real-Time Counterparts

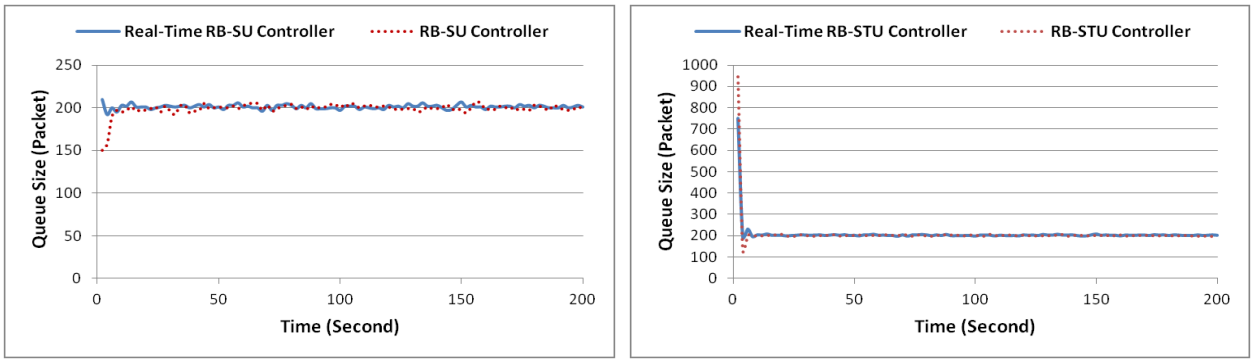
b) Queueing Delay

Fig. 6.11 shows the queueing delay of different controllers under bandwidth changes. All the controllers show smooth queueing delay in the steady state. Since the real-time RB-SU and RB-

STU controllers have smaller rise/fall time, they show smoother transition on the bandwidth step changes than the RB-SU and RB-STU controllers.

6.3.1.2 Changing Network Load Scenario

In this section, we shall investigate the performance of the real-time controllers under changing the changing network load scenario introduced in Section 4.5.2.9a. We want to study and compare the queue size and the throughput between the real-time and non-real-time controllers.



(a) Real-Time RB-SU Controller **(b) Real-Time RB-STU**

Fig. 6.12: Queue Size of the Real-Time RB-SU and RB-STU Controllers under Varying Network Load and the Comparison to Their Non-Real-Time Counterparts

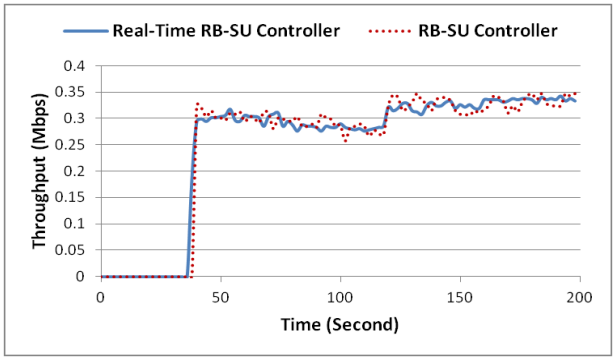
a) Queue Size

Fig. 6.12 compares the queue size of different controllers under changing *N* scenario. The real-time RB-SU controller has smaller oscillation compared to the RB-SU controller (~σ_q= 3packets for the real-time RB-SU controller compared to 7 packets for the RB-SU controller). The real-time STU controller also has smaller oscillation and peak queue value compared to the STU controller (~σ_q=23packets and peak value of 750 packets for the real-time STU controller compared to 44 packets and peak value of 956 packets for the RB-STU controller).

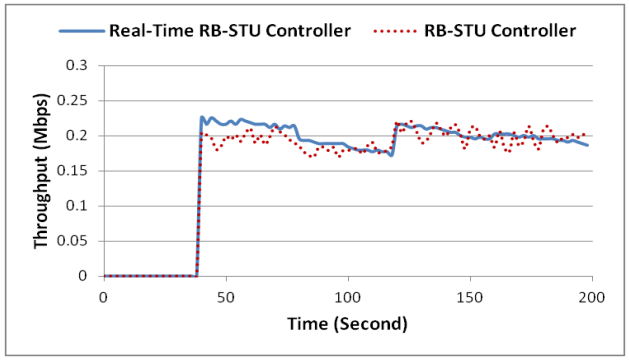
b) Throughput

Fig. 6.13 shows an example of the throughput (source sending rate) of one ftp sources (e.g. ftp 11) for different controllers under changing *N* scenario. Note that ftp 11 starts sending date from t=40 sec. All the controllers are sending data similarly in general. However, it can be seen

from the figure that the Real-Time rate-based controllers have smoother throughput than the rate-based ones.

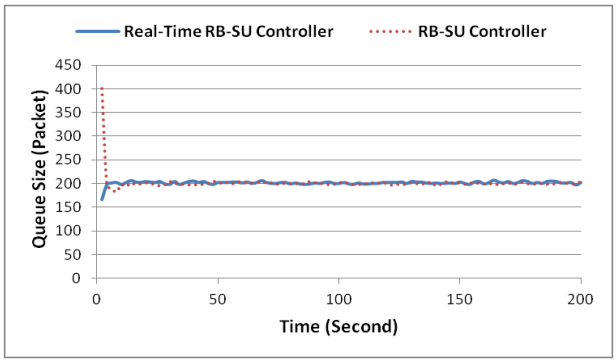


(a) Real-Time RB-SU Controller

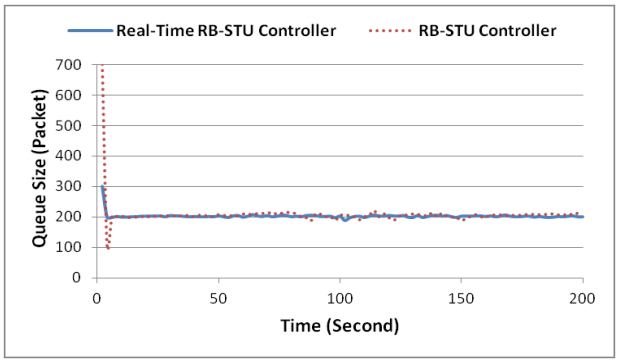


(b) Real-Time RB-STU

Fig. 6.13: Throughput of the Real-Time RB-SU and RB-STU Controllers under Varying Network Load and the Comparison to Their Non-Real-Time Counterparts



(a) Real-Time RB-SU Controller



(b) Real-Time RB-STU

6.14: Queue Size of the Real-Time RB-SU and RB-STU Controllers under Varying Network Load and Bandwidth Changes and the Comparison to Their Non-Real-Time Counterparts

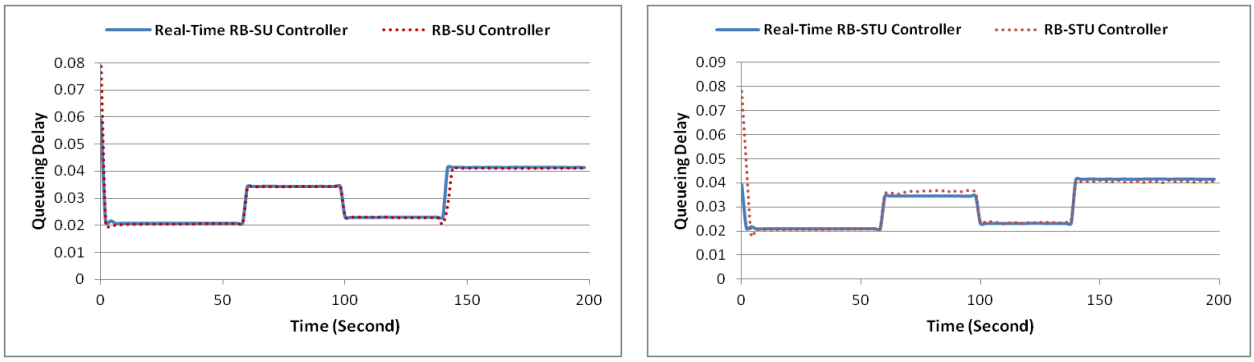
6.3.1.3 Varying Network Load and Sudden Bandwidth Changes Scenario

In this section, we shall investigate the performance of the real-time controllers when both the number of the TCP sessions and the network bandwidth are changing. We shall use the same scenario as mentioned in Section 6.2.2.3. We shall compare the queue size, queueing delay and the throughput between the real-time and non-real-time controllers.

a) Queue Size

It can be seen from Fig. 6.14 that the real-time RB-SU and RB-STU controllers have smaller peak

value and rise/fall time compared to the RB-SU and the RB-STU controllers respectively (~peak value of 205packets and T_f=3sec for the real-time RB-SU controller compared to 401 packets and T_f=12sec for the RB-SU controller and peak value of 303packets and T_f=2sec for the real-time RB-STU controller compared to 700 packets and T_f=7sec for the RB-STU controller).



(a) Real-Time RB-SU Controller **(b) Real-Time RB-STU**

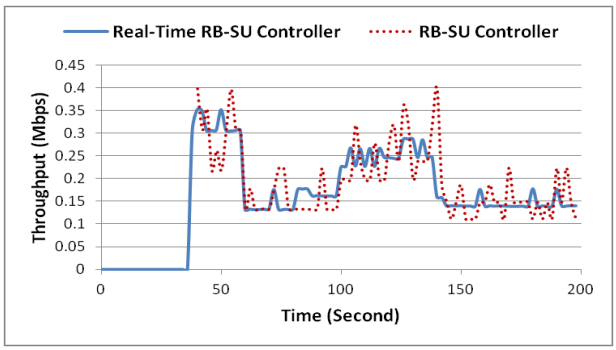
Fig. 6.15: Queueing Delay of the Real-Time RB-SU and RB-STU under Varying Network Load and Bandwidth Changes and the Comparison to Their Non-Real-Time Counterparts

b) Queueing Delay

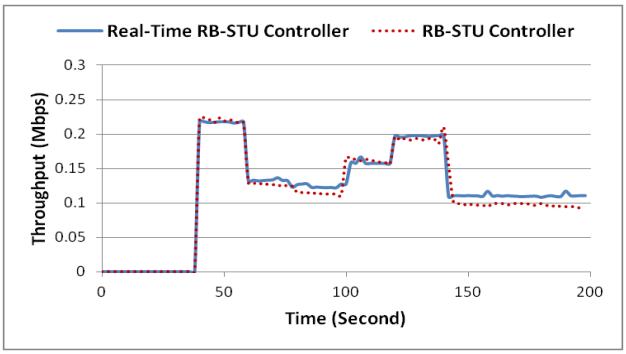
Fig. 6.15 compares the queueing delay of different controllers. One can see that all controllers show smooth queueing delay except the non-real-time RB-SU and RB-STU controllers start at higher delays compared to the real-time the RB-SU and RB-STU controller (~0.08sec for the RB-SU controller compared to 0.06sec for the real-time RB-SU controller and 0.08sec for the RB-STU controller compared to 0.04sec for the real-time RB-SU controller).

c) Throughput

Fig. 6.16 compares an example of the throughput (source sending rate) of one ftp sources (e.g. ftp 11) for different controllers. According Table 4.15, ftp 11 starts sending data from t=40 sec. It can be seen from the figure that the real-time RB-SU controller has a smoother sending rate compared to the non-real-time RB-SU controller; however, they have similar sending rate in average. One can see that the real-time RB-STU controller has slightly more sending rate than the RB-STU controller while both controllers have smooth throughput.



(a) Real-Time RB-SU Controller



(b) Real-Time RB-STU

Fig. 6.16: Throughput of the Real-Time RB-SU and RB-STU Controllers under Varying Network Load and Bandwidth Changes and the Comparison to Their Non-Real-Time Counterparts

6.4 Concluding Remarks

This chapter has proposed the real-time algorithm for H_2/H_∞ controllers using either the window-based or the rate-based operations. In this algorithm, a new controller will be generated using a built-in MATLAB process once a network parameter is changed. If the performance of the generated controller does not meet the condition of the mean and standard variation of queue length, a new controller will be generated and its performance tested again for a number of times. Our simulation results demonstrate that the real-time H_2/H_∞ controllers are robust to large network changes in network load and bottleneck bandwidth. Our real-time window-based controllers (real-time SU, ST and STU Controllers) have smaller peak queue value, less queue size fluctuations, smaller rise/fall time, smoother queueing delay and higher sending rates when compared to the non-real-time counter parts. Likewise, the real-time rate-based controllers (the real-time RB-SU and RB-STU controllers) show better performance than their non-real-time counter parts. However, the improvement is not significantly, since the rate-based controllers have already shown very smooth queue size and queuing delay with small rise/fall time. Although not shown in our documentation, we observed that the real-time window-based controllers still have more oscillations in the queue size and the throughput than the corresponding real-time rate-based controllers. However, the real-time window-based controllers have more throughput in average. Note also that although the real-time window-based controllers show better performance than the non-real-time ones, they still show poorer performance than non-real time rate-based controllers.

Chapter 7
Design Guideline

Having investigated the window-based, rate-based and real-time controllers in details, we would like to offer some recommendation/guidelines on their design, especially on any limits of the design parameters. This would also capture many repetitious results that cannot be shown for reasons of clarity and space limitation.

7.1 Guideline for Window-Based Controller Design

Based on the experiments conducted in Chapter 4, Appendices D, E and F as well as others not shown, we summarize below the ranges of parameters that give good performance of the window-based H_2/H_∞ controllers. All recommendations are based on N=100 and R_0= 0.25 and C= 9708.

7.1.1 The SU controller

Recall that we need to determine weight functions W_s and W_U for the SU controller.

7.1.1.1 Recommendation on Selecting W_s Parameters

We have determined in Section 3.2.1 that W_S takes on the function $\frac{s/M+\omega_B}{s+\omega_B A}$ so that the magnitude of the sensitivity function $|S(j\omega)|$ has an upper-bound of A at low frequency and of M at high frequency. Also selecting A<<1 ensures the approximation of $S(0) \approx 0$. The frequency ω_B can be used to approximate the bandwidth requirement [SkPo05]. This leads us to the following consideration in selecting the sensitivity weights.

From extensive simulations shown in Section 4.2.2.1, we recommend to set M between 0.7 and 2. Otherwise, a smaller or a larger M may yield a slower response. An order of 10^{-4} to 10^{-7} would be a good choice of A. We also recommend choosing ω_B within the range of 5 to 7. Increasing ω_B causes worse robustness against uncertainties.

7.1.1.2 Recommendation on Selecting W_U Parameter

According to Sections 3.2.2 and 4.2.2.2, W_U is used to bound the variation of packet drop

probability δ_p around the operating point $p_0=2N^2/(R_0\times C)^2$ [HoMi01]. In addition, the variation δ_p should be negligible when compared to p_0 in order to have a stable system. Therefore, δ_p can be chosen 100 times (or more) smaller than p_0. Hence, W_u should be chosen 100 times (or more) greater than $1/p_0$. For the example/experiment used in the text, when N=100 and R_0= 0.25 and C= 9708, $p_0 = 3.395\times10^{-3}$, and $1/p_0=2.94\times10^2$, W_U can be chosen in the order of 10^4 or more. From extensive simulations shown in Section 4.2.2.2 and Appendix D, we would recommend to set W_U between 10^4 and 0.5×10^6. Larger values of W_U usually result in worse robustness against uncertainties and longer rise/fall time.

7.1.2 The ST Controller

Recall that we need to determine weight functions W_s and W_T for the ST controller.

7.1.2.1 Recommendation on Selecting W_T Parameters

We have determined in Section 3.2.3 that $W_T(s)$ should take on the form of $\frac{a_1+a_2 s}{1+a_3 s}$ where a_1 is a small constant to ensure that $|W_T|$ is small at low frequencies. Furthermore, one would choose $a_3 << a_2$ in order to guarantee $|W_T|$ to be large at high frequencies.

From the extensive simulations shown in Section 4.3.2.1, the following guidelines are concluded regarding the choices of a_1, a_2 and a_3 for the network parameters values of N=100 and R_0= 0.25 and C= 9708

1) Since a_1 should be a small constant to ensure that $|W_T|$ is small at low frequencies, it can be chosen within the range of 0.01~ 0.0003 to have small fall/rise time.
2) Since the a_2 and a_3 should be chosen such that $a_3 << a_2$ in order to guarantee $|W_T|$ to be large at high frequencies, it is recommended that a_2 and a_3 to be set within the range of 3.5~4 and 10^{-2} respectively to have small fall/rise time.

7.1.2.2 Recommendation on Selecting W_S Parameters

Based on the results in Appendix E, we recommend choosing the W_s weight function as the one determined in Section 3.2.1 for the SU controller. One can also use the recommendation range for the W_s in Section 7.1.1.1 to define W_T without knowing the SU controller information.

7.1.3 The STU Controller

For the STU controller, we need to determine all three weight functions W_S, W_U and W_T.

7.1.3.1 Recommendation on Selecting W_T Parameters

We have determined in Section 4.4 that that $W_T(s)$ should take on the same form as in Section 3.2.3 with the same relative relationship among a_1, a_2 and a_3. From the extensive simulations shown in Appendix F, the following guidelines are concluded regarding the choices of a_1, a_2 and a_3 for the network parameters values of N=100 and R_0= 0.25 and C= 9708

1) The a_1 can be chosen within the range of 0.00031~1 which would give small $|W_T|$ at low frequencies. Beyond this range, we would encounter longer rise/fall times.
2) It is recommended that a_2 and a_3 to be set within the range of 2~7 and 10^{-2} respectively to guarantee $|W_T|$ to be large at high frequencies and having small rise/fall time

7.1.3.2 Recommendation on Selecting W_S and W_U Parameters

From simulation results mentioned in Appendix F, we suggest to use W_S and W_U parameters values as recommended for the SU controller in Section 7.1.1.

7.2 Guideline for Rate-Based Controllers

Based on the experiments conducted in Chapter 5, Appendices H and I as well as others not shown, we summarize below the ranges of parameters that give good performance of the rate-based H_2/H_∞ controllers

7.2.1 The RB-SU Controller

Recall that we need to determine weight functions W_s and W_U for the RB-SU controller.

7.2.1.1 Recommendation on Selecting W_s Parameters

From extensive simulations in Section 5.3.2.1 and Appendix H, the following guidelines are concluded the choice of M, A and ω_B.

1) set M greater than 1 because for M less than 1, the system cannot reach the desired queue size value.

2) select $A \ll 1$, in order of 10^{-2} to 10^{-3}. Lower value of A ensures better tracking performance.

3) choose ω_B within the range of 2 to 8. Increasing ω_B causes worse robustness against uncertainties. A large ω_B results in unstable queue size and a small ω_B results very long T_r.

7.2.1.2 Recommendation on Selecting W_U Parameter

According to Sections 3.3.2.2 and 5.3.2.2, W_U is used to bound the plant input which in the RB-SU controller is the variation of the advertised window size around the operating point $W_0 = \frac{R_0(C-\nu)}{N}$. In addition, the variation if the window size should be negligible when compared to W_0 in order to have a stable system. Therefore, W_u can be chosen in the order of $1/W_0$ multiplied by 10^3 or 10^4. For the example/experiment used in the text, when N=100 and R_0= 0.25 and C= 9708, $W_0 = \frac{R_0(C-\nu)}{N} = \frac{0.25\times 9708}{100} = 24.27$. Therefore, W_u should be chosen in the order of 10 or 10^2. Note that we assume that there is no uncontrolled traffic, then ν=0.

7.2.2 RB-STU Controller

For the RB-STU controller, we need to determine all three weight functions W_s, W_U and W_T.

7.2.2.1 Recommendation on Selecting W_T Parameters

We have determined in Section 5.4 that $W_T(s)$ should take on the same form as in Section 3.2.3 with the same relative relationship among a_1, a_2 and a_3. From the extensive simulations shown in Appendix I, the following guidelines are concluded regarding the choices of a_1, a_2 and a_3

1) The a_1 should be small constant to ensure that $|W_T|$ is small at low frequencies. Based on simulations, it should be chosen to be smaller than 4 due to a small $|W_T|$ at low frequencies.

2) We also need to make sure $a_3 << a_2$ in order to guarantee $|W_T|$ to be large at high frequencies. It is recommended to set a_2 a to be smaller than 4 and a_3 to be greater or equal to 0.01 in order to guarantee $|W_T|$ to be large at high frequencies.

7.2.2.2 Recommendation on Selecting W_S and W_U Parameters

From simulation results mentioned in Appendix I, we suggest to use W_S and W_U parameters values as recommended for the SU controller in Section 7.2.1.

7.3 Guideline for the Real-Time Controllers

The superior performances of the real-time H_2/H_∞ controllers have been shown in Chapter 6. The sets of weight functions have been investigated for the real-time H_2/H_∞ controllers. Below, our experiences of selecting these sets of the weight functions will be discussed.

7.3.1 Guideline for Real-Time Window-Based Controllers

We need to determine the sets of the weight functions for the real-time window-based controllers. This section suggests some guidelines to choose the parameter values of these sets. As mentioned in Chapter 6, each set of weight functions starts with the value determined in Chapter 4 for each controller. The others are the values close to that value which are also mentioned within the recommendation range mentioned in this chapter. These values are used to select (N_{loop}+1)=6 sets mentioned in Section 6.2.

7.3.1.1 The Real-Time SU Controller

This section suggests some guidelines to choose the parameter values of the W_s and W_U.

Table7.1: Weight Functions Recommended Values for the Real-Time SU Controller

$W_s = \frac{s/M + \omega_B}{s + \omega_B A}$	M	0.7, 1, 1.5, 2
	ω_B	5.5, 6, 6.5, 7
	A	10^{-7} , 10^{-6} , 10^{-5}
W_U		10^4, 8×10^4, 9×10^4 , 10^5, 2×10^5

Table 7.1 shows the recommended values for each of M, A, ω_B and W_u from Section 7.1.1 which can be used for the real-time SU controller.

7.3.1.2 The Real-Time ST Controller

Similar to the real-time SU controller, Table 7.2 below gives the recommended parameter values from Section 7.1.2 for W_s and W_T in order to design the real-time ST controller.

Table7.2: Weight Functions Recommended Values for the Real-Time ST Controller

$W_s = \frac{s/M + \omega_B}{s + \omega_B A}$	M	0.7, 1, 1.5, 2
	ω_B	5, 5.5, 6, 6.5, 7, 5
	A	$10^{-7}, 10^{-6}, 10^{-5}, 10^{-4}$
$W_T(s) = \frac{a_1 + a_2 s}{1 + a_3 s}$	a_1	0.001, 0.005, 0.01, 0.05, 0.1
	a_2	2, 3, 3.5, 4
	a_3	0.1, 0.3, 0.5, 1

Table7.3: Weight Functions Recommended Values for the Real-Time STU Controller

$W_s = \frac{s/M + \omega_B}{s + \omega_B A}$	M	0.7, 1, 1.5, 2
	ω_B	5.5, 6, 6.5, 7
	A	$10^{-7}, 10^{-6}, 10^{-5}$
$W_T(s) = \frac{a_1 + a_2 s}{1 + a_3 s}$	a_1	0.001, 0.01, 1
	a_2	2, 2.5, 3.5, 6, 7
	a_3	0.001, 0.01
W_U		$10^4, 8\times10^4, 9\times10^4, 10^5, 2\times10^5$

7.3.1.3 The Real-Time STU Controller

Similar to the real-time SU and ST controllers, Table 7.3 suggests the recommended parameters values from Section 7.1.3 for the W_U, W_s and W_T weight functions in order to design the real-time STU controller.

7.3.2 Guideline for Real-Time Rate-Based Controllers

We need to determine the recommended values of the weight functions parameters for the real-time rate-based controllers. The starting point of each set is the determined value of the weight functions in Chapter 5. The other values are chosen as close values within the recommendation range mentioned in Section 7.2. These values are used to select (N_{loop}+1)=6 sets mentioned in Section 6.3.

Table7.4: Weight Functions Recommended Values for the Real-Time RB-SU Controller

$W_s = \frac{s/M + \omega_B}{s + \omega_B A}$	M	5, 5.1, 5.2, 5.3, 5.4
	ω_B	3, 4.5, 5, 6
	A	10^{-2}, 10^{-3}
W_U		10, 20, 50, 70, 100

7.3.2.1 The Real-Time RB-SU Controller

This section suggests the guideline to choose the parameter values of the W_s and W_U for the real-time RB-SU controller. Table 7.4 shows the recommended parameters values for the W_s and W_U for the real-time RB-SU controller form Section 7.2.1.

Table7.5: Weight Functions Recommended Values for the Real-Time RB-STU Controller

$W_s = \frac{s/M + \omega_B}{s + \omega_B A}$	M	5, 5.1, 5.2, 5.3, 5.4
	ω_B	3, 4.5, 5, 6
	A	10^{-2}, 10^{-3}
$W_T(s) = \frac{a_1 + a_2 s}{1 + a_3 s}$	a_1	0.001, 0.002, 0.01, 0.1, 1
	a_2	1, 2.5, 2.75, 3, 3.5
	a_3	0.01, 0.05, 0.01
W_U		10, 20, 50, 70, 100

7.3.2.2 The Real-Time RB-STU Controller

Similar to the real-time RB-SU controller, Table 7.5 suggest the recommended parameters values for the W_s, W_U and W_T weight functions in order to design the real-time RB-STU controller from Section 7.2.2.

7.4 Concluding Remarks

In this chapter, we have shared our experiences on parameters designs of different controllers. We have given our recommendations on selecting the weight functions parameters value for the window-based controllers (the SU, ST and STU controller) as well as the rate-based controllers (RB-SU and RB-STU). We also discussed the sets of the weight functions needed for

designing the real-time controllers.

Note that our algorithm may also support gradual parameter changes by comparing the current value of the parameters (e.g., R_0) with the previous ones at regular intervals. If the difference is less than a predefined threshold value, the controller is not changed, and all network parameter values are kept for the next interval check point. Sudden changes will be monitored and with controller updated as done before. Like other algorithms, our congestion control algorithm has a linear complexity with respect to N. Therefore, there is a limit of N a given computer processor can handle in order to respond to real-time traffic. This will be commented further in the Future Work of the Conclusion chapter.

Chapter 8
Conclusion

We have put forward different controllers based on H_2/H_∞ formulation in order to address the issues of existing congestion control in Internet network. The H_2/H_∞ controller has integrated the H_2 controller and H_∞ controller in order to have good transient performance from H_2 controller and good robust property from H_∞ controller. The optimization problem formulated to design the H_2/H_∞ controller is solved by an LMI approach using MATLAB. Our designs have been used sensitivity and weight functions to monitor the closed-loop dynamics and bound them to have reasonable performances. By choosing different combinations of the sensitivity functions, we can obtain different H_2/H_∞ controllers.

The window-based H_2/H_∞ controllers are designed to control the variation of queue length in the router (in addition to maintaining the router queue length at the desired value) by controlling the packet drop probability. We have investigated the SU, the ST and the STU controllers to achieve better performance. We have shown that these controllers are stable using Lyapunov's First Method. We have verified the performance of the controllers by OPNET simulation using different performance measures of queue length, throughput, queueing delay, packet loss rate and goodput. All three controllers have shown improvement in transient response (such as rise/fall and peak queue value) and in steady state such as the small oscillation from the target queue size in comparison with the PI and the H_∞ controller. They have shown robust performance under different scenarios of bandwidth changes, having uncontrolled traffic, having different RTT, and varying *N* scenarios. For varying *N* scenario, the ST controller cannot follow the changes.

We have designed the rate-based H_2/H_∞ controllers in Chapter 5. Unlike the window-based controllers, the variation of the queue length is controlled by controlling the advertised window-size for the sources. We have investigated different combinations of weight functions to design the RB-SU and RB-STU controllers. We have demonstrated our rate-based Controllers

can effectively enhance the performance of the network of the window-based Controllers in Chapter 4. The rate-based controllers have significantly less fluctuations in queue size and queueing delay in the congested router. They have smoother sending rate (throughput) and goodput, although they have lower sending rate and goodput on average. Moreover, the rate-based controllers have shown robust performance under different scenarios of bandwidth changes, having uncontrolled traffic, having different RTT and different *N* scenarios.

Finally, we have proposed the real-time algorithm for both window-based and the rate-based controllers in Chapter 6. In this algorithm, a new controller will be generated using a built-in MATLAB process once a network parameter is changed. If the performance of the generated controller does not meet the mean and standard variation of queue length condition, another new controller will be generated and its performance tested again for a number of times. Our simulation results have demonstrated that the real-time H_2/H_∞ controllers are robust to large network changes in network load and bottleneck bandwidth. Our real-time window-based controllers (real-time SU, ST and STU Controllers) have had smaller peak queue value, less queue size fluctuations, smaller rise/fall time, smoother queueing delay and higher sending rates when compared to the non-real-time counter parts. Likewise, the real-time rate-based controllers (the real-time RB-SU and RB-STU controllers) have shown better performance than their non-real-time counter parts. However, the improvement is not significantly, since the rate-based controllers have already shown very smooth queue size and queuing delay with small rise/fall time.

Various difficulties were encountered during this research. Finding applicable/workable weight functions W_T and W_s was a time-consuming procedure as many values were not suitable for a workable controller. Among these are those for the ST controller which often have unstable/non-robust performance under the changing network load scenario. Maybe an automatic procedure could be devised. Further complication is the incompatibility of different versions of OPNET, and its support has gone downhill in recent years.

8.1 Future Work

The following are some potential and interesting aspects for extending our present work.

1) Investigating the application of the H_2/H_∞ controllers to a WLAN
2) Evaluating the performance of the H_2/H_∞ controllers under a more complex topology like a network with multiple bottlenecks. Note that the control of a single-bottleneck can be easily extended to multi-bottlenecks along a path by having the destination to send back to the sender the minimum of all advertised rates collected along the path in an ACK packet. However, the study of this scheme (or a better one) and its evaluation can be an interesting exercise.
3) Implementation of our controllers on real networks: instead of software simulations performed here, testing the controllers in real networks would be more practical to improve their deployment.
4) Analyzing the complexity of computing the H_2/H_∞ controllers in order to determine whether real-time response to traffic can be achieved according to network parameters and computing power. It may also guide us to reduce the complexity to design the H_2/H_∞ controllers such as without the need to use/call MATLAB). This would allow us to respond to different type of changes (such as gradual change in addition to sudden changes) in more advanced controller design.
5) Implementing monitors in order to obtain/estimate network parameters for the design of a controller.

References

[AbSh15] E. Abolfazli and V. Shah-Mansouri, "Robust congestion control for TCP/AQM using integral backstepping control," Proceedings of the IEEE 26th Annual International Symposium Personal, Indoor, and Mobile Radio Communications (PIMRC), Hong Kong, 2015, pp. 1840-1844.

[AlBa02] T. Alpcan and T. Basar, “A game-theoretic framework for congestion control in general topology networks”, Proceedings of the IEEE Conference on Decision and Control (CDC), pp.1218–1224, Dec. 2002.

[AlBa98] E. Altman and T. Basar, “Multiuser Rate-based flow control”, IEEE Transaction. on Communications, Vol. 46, No.7, pp.940-949, July 1998.

[AlHa09] S.M. Mahdi Alavi and M. J. Hayes “Robust active queue management design: A loop-shaping approach” Journal on Computer Communications, Vol. 32, Issue 2, 12 February 2009, pp. 324–331.

[BaEm12] S.M. Badran and A.S. Emam, "H_∞ and Mixed H_2/H_∞ with Pole-Placement Design via ILMI Method for Semi-Active Suspension System," Modeling Symposium (AMS), 2012 Sixth Asia, pp.150-155, 29-31 May 2012.

[BaRo06] A. Bacciotti and L. Rosier, Liapunov Functions and Stability in Control Theory. Springer Science & Business Media, Mar 30, 2006.

[BaSh93] R. Bambang, E. Shimemura and K. Uchida, “Mixed H_2/H_∞ control with pole placement: state-feedback case,” Proceedings of American Control Conference, pp. 2777–2779, 1993.

[BeHi06] H. Bevrani and T. Hiyama, “Robust Design of Power System Stabilizer: an LMI Approach” Proceeding of Energy and Power Systems, March20-31, Thailand.

[BoEl94] S. Boyd, L. El Ghaoui, E. Feron and V. Balakrishnan, “Linear Matrix Inequalities in System and Control Theory”, SIAM Studies in applied mathematics, Vol. 15, 1994.

[BrMa94] L. S. Brakmo, S. W. O'Malley and L. Peterson. “TCP Vegas: new techniques for congestion detection and avoidance,” Proceeding of ACM SIGCOMM '94, London, UK, pp 24-35, Aug.31-Sep 2,1994.

[Chen04] Q. Chen, “A PI Controller for IP Routers Supporting TCP Flows with Pole Placement Design” Proceeding of the Twenty-Second Biennial Symposium on Communications, pp.286-288, Kingston, Ontario, Canada June 1-3,2004.

[ChKu11] K. Chavan, R.G. Kumar, M.N. Belur and A. Karandikar, “Robust Active Queue Management for wireless Networks” IEEE Transaction on Control systems, vol. 19, No. 6, Nov. 2011.

[ChWe05] J. Cheng, D. Wei and S.H. Low” FAST TCP: from theory to experiments,” IEEE Network, vol.19, pp.4 – 11, 2005.

[ChYa03a] Q. Chen and O.W. W. Yang, “A Self-Tuning Proportional AQM Controller Based on Pole Placement” OPNETWORK 2003, August 25-29, 2003, Washington, D.C., USA, Session #: 1537, Session Name: TCP and AQM

[ChYa03b] Q. Chen and O.W.W. Yang, “On Designing a Proportional Controller for AQM Routers Based on Pole Placement,” Proceeding of the 2003 IEEE Pacific Rim

Conference on Communications, Computers and Signal Processing (PACRIM 2003), August 28-30, 2003, Victoria, B.C., Canada, pp.201-204

[ChYa04a] Q. Chen and O.W.W. Yang, “A ST-PI-PP Controller for AQM Router”, The 2004 Proceeding of the IEEE International Conference on Communications (ICC 2004), pp. 2277 – 2281, June 20-24, 2004, Paris, France.

[ChYa04b] Q. Chen and O.W.W. Yang, “On Designing Self-Tuning Controllers for AQM Routers Supporting TCP Flows Based on Pole Placement, ” IEEE Journal on Selected Areas in Communications (JSAC), Vol. 22, No. 10, Dec. 2004, pp.1965 – 1974.

[ChYa04c] Q. Chen and O. W.W. Yang, “AQM Controller Design for IP Routers Supporting TCP Flows Based on Pole Placement,” IEE Proceedings of Communications, Vol. 151, pp. 347 – 354, No. 4, Aug. 2004.

[ChYa05] Q. Chen and O. Yang “ Design of AQM controller for IP routers based on H_∞ S/U MSP” Proceeding of IEEE International Conference on Communication, ICC 2005, Vol. 1, pp. 340 – 344, 2005.

[ChYa06] Q. Chen and O. Yang, “Design of AQM controllers for IP routers based on H_∞ optimal control”, International Journal of Internet Protocol Technology, vol.1 no. 4, 2006.

[ChYa07] Q. Chen and O. Yang “Robust Controller Design for AQM Router” IEEE Transaction on Automatic Control, vol. 52, Iss. 5, pp. 938 – 943, 2007.

[DaSe08] M. Davoodi, P.K Sedgh, R. Amirifar, " H_2 and H_∞ dynamic output feedback control of a magnetic bearing system via LMIs", Proceeding of American Control Conference, pp. 2522 – 2527, 2008.

[DoBi10] R. C.Dorf and R. H. Bishop, Modern Control Systems, Prentice Hall; 12 edition, 2010.

[DuMc06] N.Dukkipati, N.McKeown and A.G.Fraser, “RCP-AC congestion control to make flows complete quickly in any environment,” Proceeding of IEEE International Conference on Computer Communication (INFOCOM) 2006, pp.1 – 5, Apr.23-29, 2006, Barcelona, Spain.

[DuNi03] X. Duan, Zh. Niu and J. Zheng, “Downlink optimization of radio resource allocation in DS-CDMA networks an economic approach”, 14" Proceeding of IEEE International Symposium on Persona1,lndoor and Mobile Radio Communication, vol.1, pp.643 – 647, 2003.

[EsSh11] A. Esmaeili and E. Shakeri, “Bacterial Foraging Optimization Robust-RED for AQM/TCP Network”, International Journal of Modeling and Optimization, Vol.1, No. 1, April 2011.

[FeVa03] W.Feng and S.Vanichpun, “Enabling compatibility between TCP Reno and TCP Vegas,” Symposium on Applications and the Internet, pp.301 – 308, 2003.

[FlJa93] S. Floyd and V. Jacobson, “Random early detection gateways for congestion avoidance”, IEEE/ACM Transactions Networking, vol. 1 no. 4, pp. 397–413, 1993.

[Floy03] S. Floyd, “High-speed TCP for large congestion windows,” RFC 3649, December 2003.

[GeCr01] P. Gevros, J. Crowcroft, P. Kirstein and S. Bhatti, “Congestion control mechanisms and the best effort service model,” IEEE Journal on Network, vol. 15, no. 3, pp.16-26, May/Jun. 2001.

[GeLo02] M. Gerla, R. Locigno, S. Mascolo and W. Weng, “Generalized Window Advertising for TCP Congestion Control”, European Transactions on Telecommunications, No. 6, pp. 549-562, Nov/Dec. 2002.

[GuYa07] X. Guan, B. Yang and B. Zhao” Adaptive fuzzy sliding mode active queue management algorithms,” Telecommunication Systems Journal, vol. 35, pp. 1-2 , June, 2007.

[HaBe07] M.M. Hassani and R. Berangi, “An analytical model for evaluating utilization of TCP Reno,” Proceeding of International Conference Computer Systems and Technologies, pp.IIIB. 14 January, Bulgaria, 2007.

[HaKa99] B. Halder and T. Kailtah, “LMI based design of mixed H_2/H_∞ controllers: The state feedback case,” Proceeding of American Control Conference, pp. 1866–1870, June 1999.

[HaKh10] M.S. haniki and M.J. Khosrowjerdi, "Multi-objective H_2/ H_∞ control design for a VSTOL flight model", Proceeding of Electrical Engineering (ICEE), 2010 18th Iranian Conference on, pp. 704 – 709, 2010.

[HeBo00] U. Hengartner, J. Bolliger and Th. Gross, “TCP Vegas revisited,” , Proceeding of IEEE International Conference on Computer Communications (INFOCOM) 2000, Tel-Aviv Vol. 3, pp.1546–1555, March 26 - 30, 2000.

[HoCh02a] Y. Hong, Q. Chen and O.W.W. Yang, “A Proportional Controller for AQM Routers Supporting TCP Flows with Simulation Using OPNET Modeler,” OPNETWORK 2002, August 26-30, 2002, Washington, D.C., USA.

[HoCh02b] Y. Hong, Q. Chen and O.W.W. Yang, “A PI Controller for AQM Routers Supporting TCP Flows,” Proceeding of IASTED International Conference on Applied Modeling and Simulation (AMS 2002), pp.157-162, Nov. 4-6, 2002, MIT, Cambridge, USA.

[HaFl03] M. Hanley, S. Floyd, J. Padhye and J. Widmer, “TCP Friendly Rate Control (TFRC): Protocol Specification”, RFC 3448, Proposed Standard, January 2003.

[HoHo03] W.K. Ho, Y. Hong, A. Hansson, H. Hjalmarsson and J.W. Deng, “Relay Auto- Tuning of PID Controllers using Iterative Feedback Tuning” Journal of *Automatica*, Vol. 39, Iss. 1**,** January 2003, pp.149-157.

[HoMi01] C.V. Hollot, V. Misra, D. Towsley and W. B. Gong, “A control Theoretic Analysis of RED” Proceeding of 20th INFOCOM01, vol.3, pp.1510-1519, 2001.

[HoMi02] C.V. Hollot, V. Misra, D. Towsley and W.B. Gong, "Analysis and Design of Controllers for AQM Routers Supporting TCP Flows," IEEE Transaction on Automatic Control, Vol. 47, pp. 945-959. Jun. 2002.

[Hoya04] Y. Hong, O.W.W. Yang and Ch. Huang “Self-Tuning PI TCP Flow Controller for AQM Routers with Interval Gain and Phase Margin Assignment”, Proceedings of IEEE Global Telecommunications Conference (Globecom 2004), Dallas, U.S.A, November 2004, pp. 1324-1328.

[HoYa06] Y. Hong and O.W.W. Yang, “Adaptive AQM controllers for IP routers with a heuristic monitor on TCP flows”, International Journal of Communication Systems 19, pp. 17–38, 2006.

[HoYa07] Y. Hong and O. W.W. Yang,” Design of adaptive PI rate controller for best-effort traffic in the Internet based on phase margin,” IEEE Transaction on Parallel and Distributed Systems, Vol. 18, Issue 4, pp.550 – 561, April 2007.

[HoYa10] Y. Hong and O. W.W. Yang," An API-RCP design using pole placement technique," Proceeding of IEEE International Conference on Communications, ICC pp.1,5, 23-27 May 2010.

[HuXi09] W. Hu and G. Xiao, "Design of congestion control based on instantaneous queue size in the routers", Proceedings of the IEEE Global Communications Conference (GLOBECOM), pp.1-6, Nov.30-Dec.04, 2009.

[HwHa05] C.L Hwang and S.Y Han, "Mixed H_2/H_∞ design for a Decentralized Variable Structure Control With Application to Mobile Robots" IEEE Transaction on Systems, Man, and Cybernetics, Vol.35, Issue 4, pp. 736 – 750, 2005.

[Jaco88] V. Jacobson, "Congestion avoidance and control," Proceeding of ACM SIGCOMM, pp. 314–329, 1988.

[JaMo09] F. Jamshidi, M.G. Moghadam and M. T H. Beheshti, "An LMI Approach to Mixed H_2/ H_∞ Synthesis via Dynamic Output-Feedback for Active Magnetic Bearing," Proceeding of International Conference on Advanced Computer Control, 2009. ICACC '09. pp.225,230, 22-24, Jan. 2009

[KaHa02] D. Katabi, M. Handley and C. Rohrs, "Congestion control for high bandwidth-delay product networks", Proceeding of SIGCOMM 2002, pp.89-102, Pittsburgh, Pennsylvania, USA., Aug. 19-23, 2002.

[KaHa02] D. Katabi, M. Handley and C. Rohrs, "Congestion Control for High Bandwidth Delay Product Networks", *Proceedings of ACM SIGCOMM*, August 2002.

[KeMa98] F. P. Kelly, A. K. Maulloo and D. K. H. Tan. "Rate control for communication networks: shadow prices, proportional fairness and stability," Journal of the Operational Research Society, vol. 49, no.3, pp. 237–252, March 1998.

[KeRa10] F. Kelly and G. Raina, "Explicit Congestion Control: charging, fairness and admission management" In Next-Generation Internet Architectures and Protocols Journal, Cambridge University Press, pp. 257-274, 2011

[KhRo91] P. P. Khargonekar and M. A. Rotea, "Mixed H_2/ H_∞ control: A convex optimization approach," IEEE Transaction Automatic Control, vol. 39, pp. 824–837, 1991.

[LeMo01] I.K. Leung and J.K. Muppala, "Packet marking strategies for Explicit Congestion Notification (ECN)," Proceeding of IEEE International Conference Performance, Computing and Communications, pp.17-23, 2001, Albuquerque, USA.

[Libe03] D. Liberzon, *Switching in Systems and Control*, Boston: Birkhauser, 2003.

[LiPe13] Ch. Lin and Ch. Peng, "Mixed H_2/ H_∞ control Output Feedback Stability Control for Dual-Motor Independent Drive Electric Vehicle", Journal on Advanced Material Research, Vol. 658, pp. 602-608.

[LiYa11] J. Liu and O. W.W. Yang, "Stability analysis and evaluation of the IntelRate controller for high-speed heterogeneous networks," Proceedings of the IEEE International Conference on Communications, (*ICC2011*), Kyoto, Japan, Jun. 5-9, 2011.

[LoAn05] S.H. Low, L.L.H. Andrew and B.P. Wydrowski, "Understanding XCP: equilibrium and fairness", Proceedings of IEEE INFOCOM, March 2005, pp. 1025-1036.

[LoLa99] S. H. Low and D. E. Lapsley, "Optimization flow control, I: Basic algorithm and convergence," IEEE/ACM Transaction on Networking, vol. 7, pp. 861–874, 1999.

[LoPa02a] S. H. Low, F. Paganini, and J. C. Doyle, "Internet congestion control," IEEE Contr. Syst. Mag., vol. 22, pp. 28-43, Jan. 2002.

[LoPa02b] S. Low, F. Paganini, J. Wang, S. Adlakha and J.C. Doyle, "Dynamics of TCP/RED and a Scalable Control," Proc. IEEE INFOCOM, pp. 239-248, June 2002.

[MaBe09] Sabato Manfredi, Mario di Bernardo and Franco Garofalo, "Design, validation and experimental testing of a robust AQM control" Journal on Control Engineering Practice, Vol. 17, Issue 3, March 2009, pp. 394–407

[Mack04] U. Mackenroth, *Robust Control Systems*, Springer 2004

[Masc99] S. Mascolo, "Congestion control in high-speed communication networks using the Smith principle", Journal on Automatica no. 35, pp. 1921–1935, 1999.

[Matl13] MATLAB Products, http://www.mathworks.com. Last access on Dec 01 2013.

[MiGo00] V. Misra, W. Gong and D. Towsley, "Fluid-based analysis of a network of AQM routers supporting TCP flows with an application to RED", Proceeding on SIGCOMM, pp. 151-160, 2000.

[NoGa94] A. Nomirovskii and P. Gahinet, "The Projective for Solving Linear Matrix Inequalities" Proceeding of the American Control Conference, vol.1, pp.840-844, June 29-July 1, 1994, Baltimore, Maryland.

[Ogat02] K. Ogata, "Modern Control Engineering", Prentice Hall, 4th edition, 2002.

[Opne13] OPNET Modeler Version 17.5, OPNET Technologies, Inc., URL: http://www.riverbed.com/products/performance-management-control/opnet.html, 2013.

[OtLa99] T.J. Ott, T.V Lakshman, and L.H Wong, "SRED: stabilized RED," in Proceedings of IEEE/INFOCOM, pp. 1346 –1355, 1999.

[PaGa05] W. Paszke, K. Gakowski, E. Rogers, A. Kummert and D.H Owens, "Mixed H_2/H_∞ and robust control of differential linear repetitive processes", Proceeding of Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on, pp. 7900 – 7905, 2005.

[PiPr06] A. Pietrabissa, F. D. Priscoli, A. Fiaschetti, and F. Di Paolo, "A robust adaptive congestion control for communication networks with time varying delays," Proceeding of IEEE International Conference Control Application, Munich, Germany, 2006, pp. 2093–2098.

[PoDh06] J.Polk and S.Dhesikan, "A Resource Reservation Protocol (RSVP) extension for the reduction of bandwidth of a reservation flow", RFC 4495, May, 2006.

[QaZn09] I.A. Qazi, T. Znati and L.L.H Andrew," Congestion control using efficient explicit feedback," Proceeding INFOCOM 2009, pp.10 – 18, Apr. 19-25, 2009, Rio De Janeiro, Brazil.

[QuOz04] P.F. Quet and H. Ozbay, "On the Design of AQM Supporting TCP Flows Using Robust Control Theory" IEEE Transaction On Automatic Control, vol. 49, no. 6, pp. 1031-1036, 2004.

[QuRa01] P.F. Quet, S. Ramakrishnan and H. Ozbay, "On the H_∞ controller design for congestion control in communication networks with a capacity predictor" Proceeding of IEEE Conference on Decision and Control, pp. 598 – 603, 2001.

[RaFl99] K.K. Ramakrishnan and S. Floyd, "Proposals to add explicit congestion Notification (ECN) to IP", RFC 2481, Jan.1999.

[RoHa77] N. Rouche, P. Habets and M. Laloy, “Stability Theory by Liapunov’s Direct Method,” Springer-Verlag New York INC.,1977.

[Sche95] C. W. Scherer, “Multi-objective H_2/H_∞ controller ”, IEEE Transaction on Automatic Control, vol. 40, no. 6, June 1995.

[SkPo05] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control, Analysis and Design*, John Wiley & Sons, 2005.

[Stid04] Sh. Stidham, “Pricing and congestion management in a network with heterogeneous users”, IEEE Transaction on Automatic Control pp. 976 – 981, 2004.

[TeSz05] M. Tekala and R. Szbo, “Evaluation of scalable TCP,” Proceeding of 3rd ACS/IEEE International Conference on Computer Systems and Applications, pp.55-62, 2005.

[ToEl11] I. Tolaimate and N. Elalami, “Robust Control Problem as H_2 and H_∞ control problem applied to the robust controller design of Active Queue Management routers for Internet Protocol”, International Journal of Systems Applications, Engineering and Development, Vol. 5, Iss. 6, 2011.

[TuBa07] M. C. Turner and D. Bates, Mathematical Methods for Robust and Nonlinear Control: EPSRC Summer School, Springer Science & Business Media, 2007.

[Vinn01] Gl. Vinnicombe, Uncertainty and Feedback: H_∞ loop-shaping and ν –gap Metric, World Scientific, Jan 1, 2001.

[WeJi07] D. X. Wei, Ch. Jin and S. H. Low, “Fast TCP: motivation, architecture, algorithms, performance,” IEEE/ACM Transaction On Networking, vol.14, Iss. 6, pp. 1249-1256, 2007.

[WyAn03] B. Wydrowski, L.L.H. Andrew and M. Zukerman, "MaxNet: a congestion control architecture for scalable networks", IEEE Communications Letters, vol. 7, Iss. 10pp.511 – 513, Oct. 2003.

[XiYa05] B. Xie and B. Yao, “Multi-objective optimization of tip tracking control using LMI”, Proceeding of International Conference on Mechanical Engineering Congress and Exposition, pp. 1-10, 2005, Orlando, Florida.

[YaXi06] R. Yang, L. Xie and C. Zhang, “H_2 and mixed H_2/H_∞ control of two dimensional systems In Rosser model”, Journal on Automatica, 2006, Vol. 42, pp. 1507-1514.

[ZaAm08] N.M. Zadeh and R. Amirifar, "H_∞ and Mixed H_2/H_∞ Control of Dual-Actuator Hard Disk Drive via LMIs", Proceeding of IEEE Conference on Robotics, Automation and Mechatronics, pp. 61 – 65, 2008.

[ZhAh05] Y. Zhang and M. Ahmed, "A control theoretic analysis of XCP," in Proceeding of IEEE INFOCOM , Mar. 2005.

[ZhHe05] Y. Zhang and T.R. Henderson, "An implementation and experimental study of the explicit control protocol (XCP)," Proceeding of IEEE International Conference on Computer Communication, INFOCOM, Mar. 2005.

[ZhHo03] H. Zhang, C.V. Hollot, D. Towsley and V. Misra, “A Self-Tuning Structure for Adaptation in TCP/AQM Networks”, Proceedings of IEEE Globecom 2003, pp. 1346 –1355.

[ZhLe06] Y. Zhang, D. Leonard and D. Loguinov, “JetMax: scalable max-min congestion control for high-Speed heterogeneous networks”, Proceeding of IEEE International Conference on Computer Communication, INFOCOM 2006, Apr. 23-29, pp.1-13, Barcelona, Spain, 2006.

[ZhNe07] F. Zheng and J. Nelson," an H_∞ approach to congestion control design for AQM routers supporting TCP flows in wireless access networks", Journal on Computer Networks vol. 51, Iss. 6, pp. 1684-1704, 2007.

[ZhNe09] F. Zheng and J. Nelson "An H_∞ approach to the controller design of AQM routers supporting TCP flows", Automatica Journal Vol. 45, pp. 757-763, 2009.

Appendix A
TCP Congestion Control System

This appendix summarizes the TCP congestion control operation used in this thesis as well as the TCP/AQM model used to design the window-based H_2/H_∞ Controller in Chapter 4.

A.1 TCP Operation

We summarize the TCP congestion control operation used in this thesis. Details can be found in the vast literature such as [AlPa99] and textbooks. TCP is a connection-oriented end-to-end transport protocol. TCP consider any damaged or lost packet as a failure and provides ACK for each packets received by receiver. Since TCP traces all the packets based on their sequence numbers, it is easy to reorder the packets that may be misordered during the transmission or delete the duplicate packets. TCP provides congestion avoidance algorithm to control congestion in the network. Major TCP congestion avoidance controller consist four algorithms, *Slow Start, Congestion Avoidance, Fast Retransmit* and *Fast Recovery.*

Slow Start and *Congestion Avoidance* are basic algorithms of congestion avoidance algorithms to control amount of data sent to the network. The congestion window (*cwnd*) is a sender side limit on the numbers of packets the source host can send into network before receiving ACK. Another important variable, the slow start threshold (*ssthresh*), determines whether slow start is over or congestion avoidance algorithm begins, as discussed in detail below. Since there is no information of the network at the beginning of the transmission, TCP should start transmitting slowly to see the capacity of the network.

We used Reno TCP congestion avoidance algorithms in our simulations. In Reno, the Slow Start state occurs whenever a sender starts to transmit a packet whether a new connection is established or a connection is reestablished after idle time or timing out. The *cwnd* is set to 1 at the beginning and gets doubled when the sender receives ACK of the new packet from the receiver. The sender makes the *cwnd* double until *cwnd* reaches *ssthresh* or congestion is observed. Therefore, in *Slow Start* algorithm *cwnd* is always less than *ssthresh*. In either case, *Congestion Avoidance* algorithm takes over the control of transmission. In Congestion

Avoidance algorithm, sender increases *cwnd* by one packet upon receiving a new ACK of received packet from receiver. As long as sender does not observe duplicate ACK of a packet from receiver, *cwnd* is increased. Upon sender observes three duplicate ACKs of a packet, considers a packet lost and will halve the congestion window and set *ssthresh* to new congestion window, *cwnd*, and start *Fast Retransmit* algorithm. In *Fast Retransmit*, the sender retransmits the lost packet and start *Fast Recovery* algorithm. In this state, the sender sends the lost packet and waits for the ACKs for entire transmit window before returning to *Congestion Avoidance*. If there is no ACK, Reno considers it as a timeout and enters the *Slow Start* state.

A. 2 The TCP/AQM System Model

This section provides the background of the linear model discussed in Chapter 4. The linear model presented in Section 4.1 is obtained from a nonlinear TCP/AQM model, which can be described by the following coupled, nonlinear differential equations [HoMi02]:

$$\dot{W}(t) = \frac{1}{R(t)} - \frac{W(t)W(t-R(t))}{2R(t-R(t))}\, p(t-R(t)) \qquad \text{(A.1)}$$

$$\dot{q} = \frac{W(t)}{R(t)} N(t) - C \qquad \text{(A.2)}$$

where $\dot{x}$ denotes the time-derivative of x and

W $\doteq$ TCP window size (packets);

q $\doteq$ queue length (packets);

R $\doteq$ round trip time $= \frac{q}{C} + T_p$ (secs);

T_p $\doteq$ propagation delay (secs);

C $\doteq$ link capacity (packets/sec);

N $\doteq$ load factor (number of long-lived TCP sessions);

p $\doteq$ probability of packet drop.

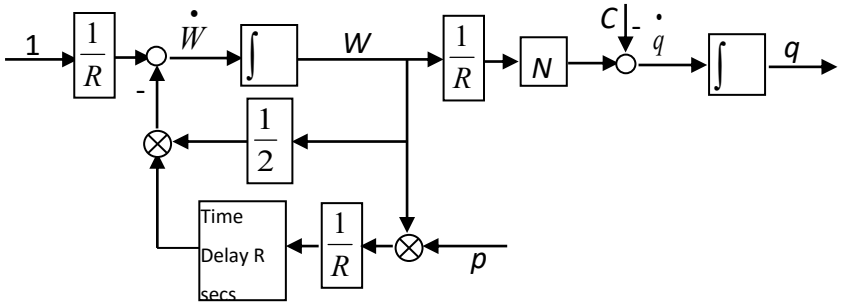


Fig. A.1: Block–diagram of the nonlinear dynamic model for TCP

The differential equations are illustrated in the block diagram of Fig A.1 that highlights the TCP window-control and queue dynamics. Eqn. (A.1) models the additive-increase multiplicative-decrease behavior of TCP. The first term corresponds to the additive increase part, which says that the window size will increase by one every round trip time. The second term corresponds to the multiplicative decrease part, which halves the window size at the instant of the arrival of a loss. Eqn. (A.2) models the bottlenecked queue dynamic. The first term corresponds to the increase in the queue length due to the arrival of packets from the *N* TCP flows. The second term models the decrease in the queue length due to the servicing of packets.

To linearize eqns. (A.1), (A.2), we first assume that the number of TCP sessions and link capacity are constant, i.e., *N(t)=N, C(t)=C*. Then by taking (*W, q*) as the state and *p* as input, the operating point (W_0, q_0, p_0) can then defined via $\dot{W}(t) = 0$ and $\dot{q} = 0$ because

$$\dot{W} = 0 \Longrightarrow W_0^2 p_0 = 2 \qquad \text{(A.3)}$$

$$\dot{q} = 0 \Longrightarrow W_0 \ = \frac{R_0 C}{N}; R_0 = \frac{q_0}{C} + T_P \qquad \text{(A.4)}$$

where R_0 is defined as the average value of RTT. Note that many works have shown reasonable results by assuming a constant R_0. By linearizing eqns. (A.1), (A.2) about the operating point (See [HoMi02] for more details) to obtain

$$\delta\dot{W}(t) = -\frac{N}{R_0^2 C}\big(\delta W(t) + \delta W(t - R_0)\big) - \frac{1}{R_0^2 C}\big(\delta q(t) - \delta q(t - R_0)\big) - \frac{R_0 C^2}{2N^2}\delta p(t - R_0)$$

$$\delta\dot{q}(t) = \frac{N}{R_0}\delta W(t) - \frac{1}{R_0}\delta q(t) \qquad \text{(A.5)}$$

where

$\delta q \stackrel{\text{def}}{=} q - q_0$

$\delta W \stackrel{\text{def}}{=} W - W_0$

$\delta p \stackrel{\text{def}}{=} p - p_0$

are the perturbed variables about the operating points.

We can now have the TCP/AQM plant model as $P(s) = \frac{\delta q}{\delta p}$ as follows. For more details, please refer to [HoMi02].

$$P(s) = \frac{\frac{-C^2}{2N}s + \frac{C^2}{R_0 N}}{s^3 + \left(\frac{2N}{R_0^2 C} + \frac{3}{R_0}\right)s^2 + \left(\frac{6N}{R_0^3 C} + \frac{2}{R_0^2}\right)s + \frac{4N}{R_0^4 C}} \qquad \text{(A.6)}$$

Appendix B

The H-Space and its Norms [SkPo05]

Norms are functions that can provide a number. A norm of *e*, denoted as $\|e\|$, is a real number which gives an overall measure of the size of *e* (which may be a matrix, signal or system). A norm satisfies the basic properties of non-negativity, homogeneity (i.e., $\|\alpha.e\| = |\alpha|.\|e\|$) and triangle inequality ($i.e.$, $\|e_1 + e_2\| \leq \|e_1\| + \|e_2\|$).

B.1. Norms of vectors

The general norm of a vector *a* is the *p*-norm defined to be $\|a\|_p = (\sum_i |a_i|^p)^{1/p}$ where we must have *p*>1 to satisfy the triangular inequality. There are three common sub-cases of the p-norm:

a) The 1-norm when *p*=1, and we have $\|a\|_1 = \sum_i |a_i|$.

b) The 2-norm (Euclidean norm) is the most common vector norm and corresponds to the shortest distance between two points. That is $\|a\|_2 = \sqrt{\sum_i |a_i|^2}$.

c) The ∞-norm is the largest-element magnitude in the vector which can be defined as $\|a\|_\infty = \max_i |a_i|$.

B.2 The Hardy Space

In control engineering, a system is usually represented by a transfer function *E(s)* relating the output to its input signals. The Hardly space of dimension-p refers to those systems with their *p*-norm defined by $\|\boldsymbol{E}(s)\|_p = (\int_{-\infty}^{+\infty} |f(j\omega)|^p d\omega)^{1/p}$ where *f(s)* is scalar transfer function where *s*=*j*ω and ω is frequency. Therefore, H_2 stands for the Hardly space whose transfer functions are bounded in the 2-norm defined by $\|\boldsymbol{E}(s)\|_2 = (\frac{1}{2\pi}\int_{-\infty}^{+\infty} |f(j\omega)|^2 d\omega)^{1/2} = \frac{1}{2\pi}\int_{-\infty}^{+\infty} (\sum_{i=1}^{m} \left(\sigma_i\left(\boldsymbol{E}(j\omega)\right)\right)^2 d\omega)$. The factor $1 / \sqrt{2\pi}$ is introduced to get consistency with the 2-norm. H_∞ space considers peak values in the maximum singularities $\bar{\boldsymbol{\sigma}}$ of $\boldsymbol{E}$. That is, $\|\boldsymbol{E}(s)\|_\infty = \max_\omega \bar{\sigma}\left(\boldsymbol{E}(j\omega)\right)$ which is maximum of singular value.

Appendix C
Upper Bounds of Sensitivity functions in SU and ST Controllers

This appendix first presents the proof [SkPo05] related to the upper bounds of $|S|$ and $|U|$ in the *SU* controller. Proofs of other controllers are similar. Readers can refer to [SkPo05] for more details. Some examples of selected weight functions and sensitivity functions of the designed SU and ST controllers are given.

C.1 Proof of the Existence of upper bounds of $|S|$ and $|U|$ in the SU Controller

Lemma: In the *SU* controller defined by Section 4.2, if K is the controller used in the system, then absolute value of the sensitivity and input sensitivity functions noted by $|S|$ and $|U|$ are upper bounded by $\gamma_{\min}/|W_S|$ and $\gamma_{\min}/|W_U|$ respectively, where $\gamma_{\min}$ is the optimal value of $\|H\|_\infty$ when K is used.

Proof:

When K is used, we have, $\gamma_{\min} = \|H\|_\infty$

$$= \max_{\omega} \bar{\sigma}(H(j\omega))$$

$$= \max_{\omega} (\sqrt{|W_S S|^2 + |W_U U|^2}) \ ,$$

which means,

$$\sqrt{|W_S S|^2 + |W_U U|^2} \le \gamma_{\min} \ , \quad \forall \omega$$

$$\Leftrightarrow \quad |W_S S| \le \gamma_{\min} \ , \quad |W_U U| \le \gamma_{\min} \ , \quad \forall \omega$$

$$\Leftrightarrow \quad |S| \le \frac{\gamma_{\min}}{|W_S|} \ , \quad |U| \le \frac{\gamma_{\min}}{|W_U|} \ , \quad \forall \omega$$

Therefore, $|S|$ and $|U|$ are upper bounded by $\gamma_{\min}/|W_S|$ and $\gamma_{\min}/|W_U|$ respectively.

C.2 Examples for the SU Controller

In this section, we present the Bode diagrams of the sensitivity and weight functions for the SU

controller.

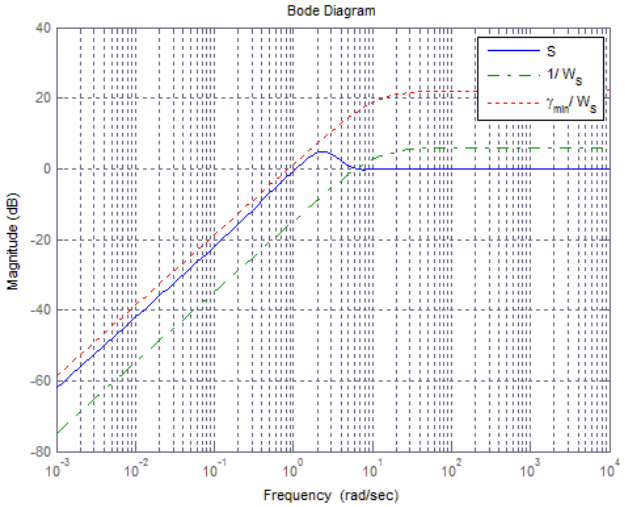


Fig C.1: Bode Diagram of S, $1/W_S$ and γ_{min}/W_S for the SU Controller

Fig C.1 presents the Bode diagrams of the sensitivity function *S(s)* (indicated by the solid blue curve), the inverse of the sensitivity weight function $1/W_S$ (the dotted red curve) and γ_{min}/W_S (the dashed green curve) of the SU controller in order to verify the feasibility of our weight functions selected for $W_s(s)$ (see Section 4.2.1.1). It is shown that *S(s)* is small at low frequencies, which means the sensitivity of the system output to disturbance is small at low frequencies, and likewise the error between system output and the reference input is small. Conversely, in order to attenuate the measurement noise and reduce the influence on the tracking error, we desire *S(s)* to be around 1 at high frequencies which means the magnitude of the open loop transfer function is large at high frequencies. The figure shows that the magnitude of *S(s)* reaches the asymptotic value of 1 (0dB) at high frequencies which is the desirable characteristic of sensitivity function [DoBi10]. In addition, Fig C.1 has verified that $|S(j\omega)|$ is upper bounded by $\gamma_{min}/|W_S|$ which meets one of the requirements in designing SU controllers.

Similarly, Fig. C.2 below represents the Bode diagrams of the input sensitivity function *U(s)*, the inverse of the sensitivity weight function $1/W_U$ and γ_{min}/W_U. It is shown that $|U(j\omega)|$ is upper bounded by $\gamma_{min}/|W_U|$, and hence the correctness of the parameter selection is verified for W_u.

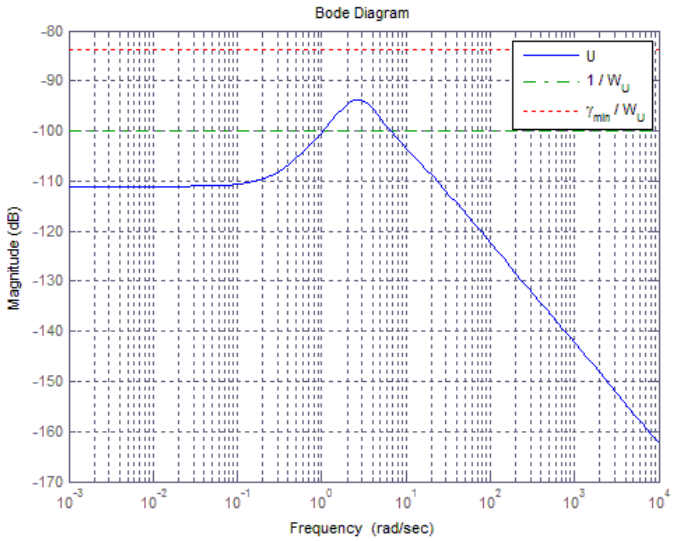
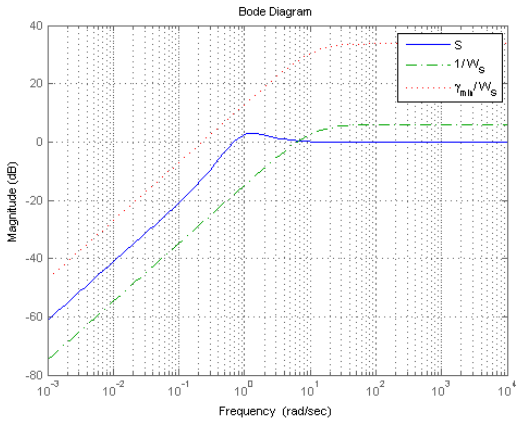


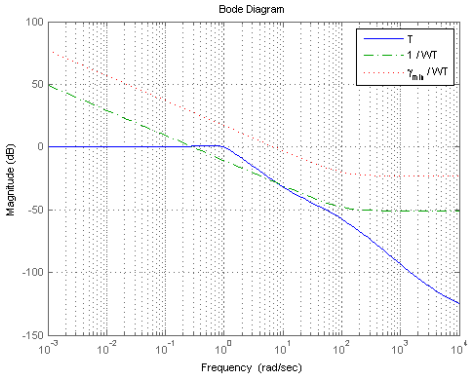
Fig. C.2: Bode Diagram of U, $1/W_U$ and γ_{min}/W_U for the SU Controller

C.3 Example for the ST Controller

The same proof as Section C.1 can be provided for the ST controller by substituting input sensitivity function U and associated weight function W_U by complementary sensitivity function T and associated weight function W_T respectively. For avoiding the repetition, we do not provide it here. In this section, we present the Bode diagrams of the sensitivity and weight functions for the ST controller. Fig C.3a shows the Bode diagrams of the sensitivity function $S(s)$ (indicated by the solid blue curve), the inverse of the sensitivity weight function $1/W_S$ (the dotted red curve) and γ_{min}/W_S (the dashed green curve) of the ST-controller in order to verify the feasibility of our weight functions selected for $W_s(s)$ (to see in Section 4.3.1.1). It is shown that $S(s)$ is small at low frequencies, which means the sensitivity of the system output to disturbance is small at low frequencies, and likewise the error between system output and the reference input is small and conversely 0 dB at high frequencies which means noise measurement is attenuated the influence on the tracking error is reduced



(a) S, $1/W_S$ and γ_{min}/W_S



(b) T, $1/W_T$ and γ_{min}/W_T

Fig C.3: Bode Diagram for the ST Controller

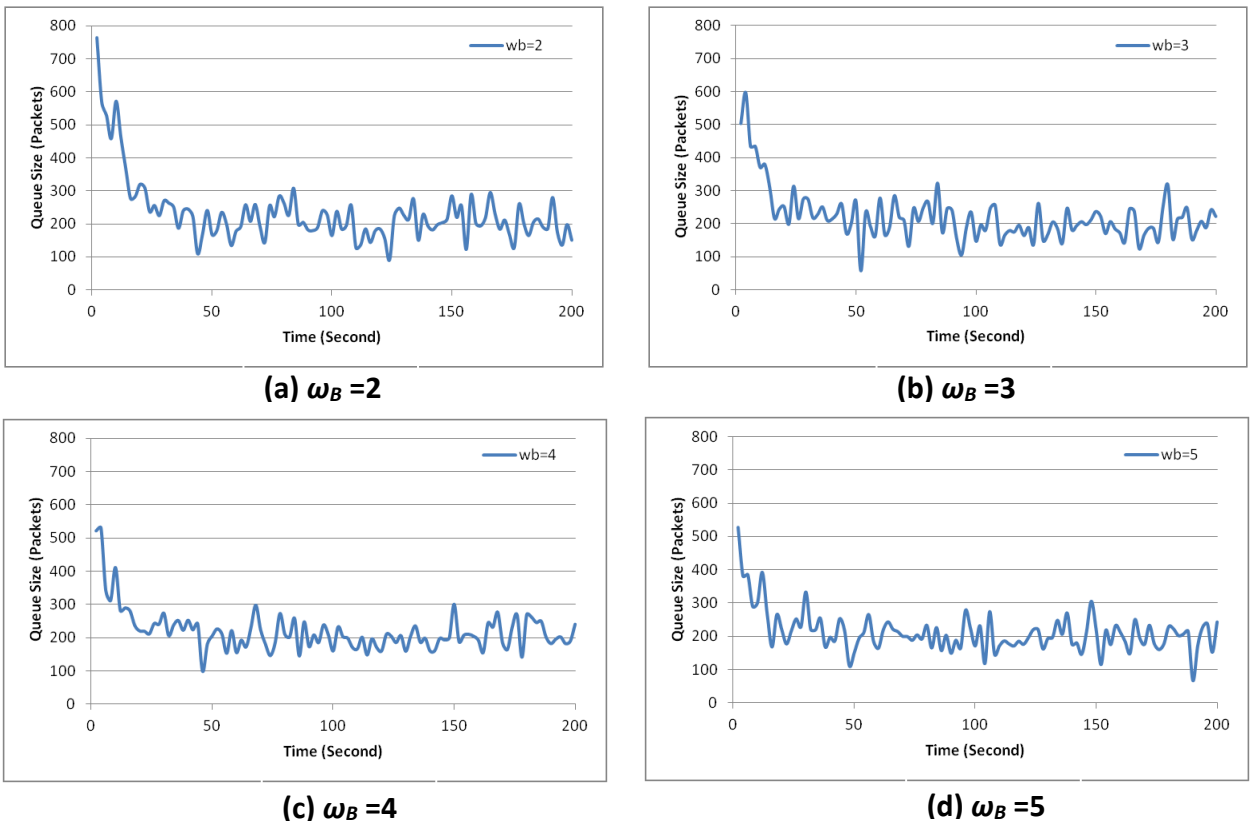
Fig C.3a has verified that $|S(j\omega)|$ is upper bounded by $\gamma_{min}/|W_S|$ which meets one of the requirements in designing *ST* controllers. Fig C.3b presents the Bode diagrams of the complementary sensitivity function T, the inverse of the complementary sensitivity weight function $1/W_T$ and γ_{min}/W_T. It is shown that T is 0 dB at low frequencies, and drops off at high frequencies, which reduces the sensitivity to noise and uncertainty at high frequencies. In addition, we have mentioned that, $|T|$ is upper bounded by $1/|W_T|$. Fig. C.3.b also verifies that $|T|$ is upper bounded by $\gamma_{min}/|W_T|$.

Appendix D
More Experiments on Selecting the SU Controller Weights Parameters

In order to show the adequacy of the proposed SU controller, more experiments on the selecting the weight parameters have been done even though not all the cases can be simulated. This appendix documents couple more selected experiments using some other weight functions parameters. In the first experiment, we shall investigate the effect of varying ω_B when other parameters are fixed at values that are different from those used in Section 4.2.2. In the second experiment, the designed controller in Section 4.2.2 is compared to other SU controllers with different weight functions values.

D.1 First Experiment

In this experiment, we shall vary ω_B while M, A and W_U are fixing at 0.2, 10^{-4} and 10^6 instead of 1.5, 10^{-5} and 10^5 determined in Section 4.2.2. Fig D.1 shows the instantaneous queue size for the SU controller simulated in OPNET for ω_B ranging from ω_B =2 and 5. In comparison to Fig. 4.6, all controllers designed here have longer rise/fall time and peak value. To have deeper insight, Table D.1 gives different performance measures including T_r , peak value, μ_q, σ_q as a function of ω_B while fixing M, A and W_U at 1.5, 10^{-5} and 10^5 respectively. One can see that the designed controllers here have significantly longer T_r than the controllers proposed in Table 4.6. The longest and shortest T_r are achieved when ω_B =2 and 5 (~35 sec and 15 sec respectively) in compare to 14 sec and 6 sec in Table 4.6. In addition, for the designed controllers here, the largest and smallest peak queue size value are 763 and 534 packets which significantly larger than the proposed controllers in Table 4.6 (~384 and 296 packets). Due to having large queue size at the beginning, mean queue length and the standard deviation in this case are also larger than the proposed controllers in Table 4.6. Please note that due to avoid repetition, we only show this scenario. The same experiments can be investigated for varying M, A, and W_u.



(a) ω_B =2 (b) ω_B =3

(c) ω_B =4 (d) ω_B =5

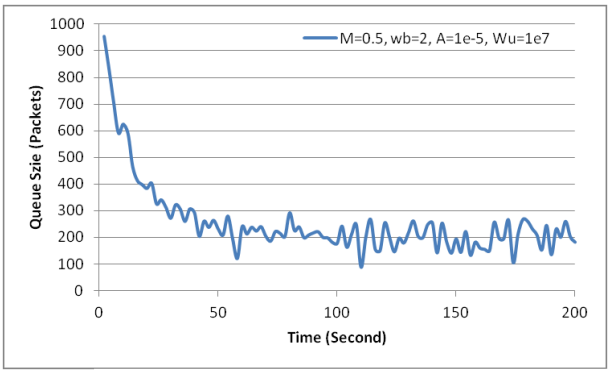
Fig. D.1: Instantaneous Queue Size of the *SU* Controller with W_U =10^6, A= 10^{-4} and M= 0.2 but Different ω_B Values

Table D.1: Performance of SU Controllers with Different ω_B

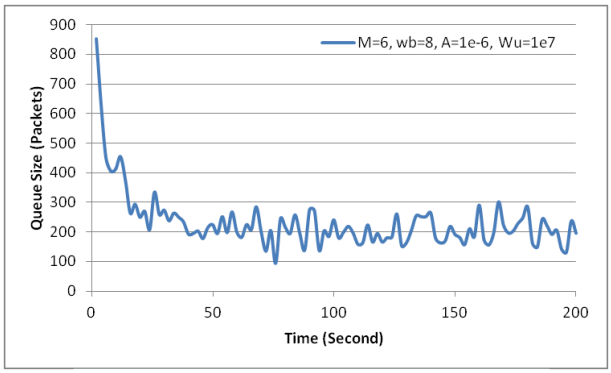
W_U=10^6, A=10^{-4}, M=0.2	T_r (Second)	Peak Value (Packet)	μ_q (Packet)	σ_q(Packet)
ω_B =2	35	763	233	98
ω_B =3	21	598	221	77
ω_B =4	23	524	220	63
ω_B =5	15	531	212	61

D.2 Second Experiment

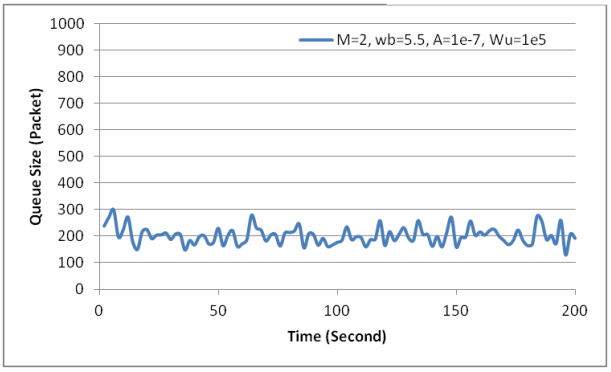
Now, we shall compare the proposed SU controller in Section 4.2.2 to another SU controllers designed by different weight function parameters values designed for the same system parameters which are C=9708, N=100, R_0=0.25 second.



(a) M=0.5, ω_B =2, A=10^{-5}, W_U=10^7



(b) M=6, ω_B =8, A=10^{-6}, W_U=10^7



(c) M=2, ω_B =5.5, A=10^{-7}, W_U=10^5

Fig D.2: Comparison of Instantaneous Queue Size for Different SU Controllers

Fig D.2 shows different SU controllers designed for the same system parameters. Fig D.2a shows the queue size for the SU controller with $W_s(s) = \frac{2s+2}{s+2\times10^{-5}}$ and W_U =10^7, Fig D.2b with $W_s(s) = \frac{s+48}{s+48\times10^{-6}}$ and W_U =10^7 and Fig D.2c $W_s(s) = \frac{s+11}{2s+11\times10^{-7}}$ and W_U =10^5 (the proposed controller in Section 3.2.1.1). On can see that our proposed controller has significantly better performance with shorter T_r (6 sec to 41s, 38s for controllers in (a) and (b) respectively) and smaller peak value (290 packets to 956 and 856 packets for controllers (a) and (b) respectively).

Appendix E
More Experiments on Selecting the ST Controller Weight Parameters

In order to show the adequacy of the proposed ST controller in Section 4.3, more experiments on selecting the W_s weight function (when W_T is fixed) are given in this appendix.

E.1 Effect of Varying W_S

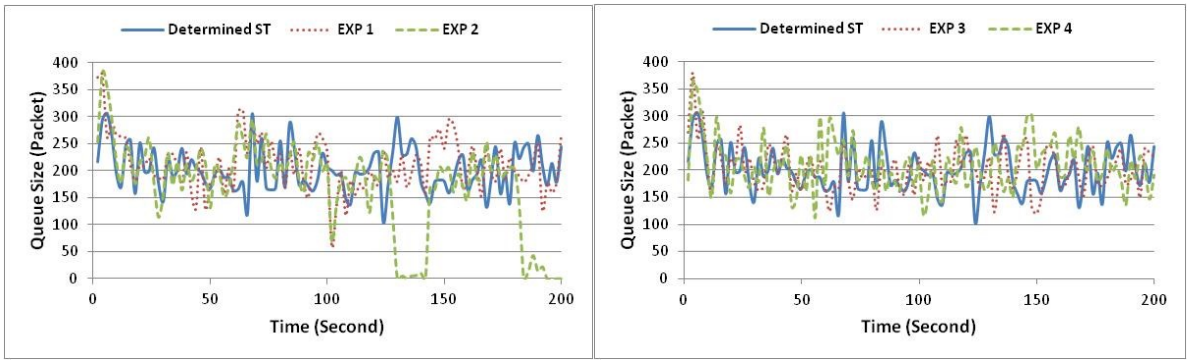
In this section, we shall investigate the effect of varying W_s while W_T is fixed. Specifically, we vary the parameters ω_B, M or A *in* W_S while fixing W_T at $W_T(s) = \frac{0.00031+3.5s}{1+0.01s}$ as determined in Section 4.3.2.1. Note that we shall use the same example as in Section 4.2.2.1.

Table E.1: Different Weight Functions for More Experiments for the ST controller

$W_T(s) = \frac{0.00031 + 3.5s}{1 + 0.01s}$	$W_s(s) = \frac{s/M + \omega_b}{s + \omega_b \times A}$		
	M	ω_b	A
Determined ST	2	5.5	10^{-7}
EXP 1	4	5.5	10^{-7}
EXP 2	1	5.5	10^{-7}
EXP 3	2	6	10^{-7}
EXP 4	2	5.5	10^{-5}

Table E.1 shows different experiments when W_s parameters is varying. We change different W_s parameters value one at a time. In experiments 1 and 2, the parameters M is varying when ω_B, A and W_T are fixed. In experiment 3 and 4, the parameters value of ω_B and A are different from the determined ST controller respectively. We have experimented wide ranges of different parameters values. However, as mentioned in Sections 4.3.2 and 4.3.2.2, it is difficult to find weight parameters values, which the ST controller can have valid/operative performance. We only shows the parameters values which the ST controller performs reasonable. For many

parameters values, the ST controller is not able to perform and the router 1 queue size is zero. For example we found that when [M, ω_B, A]= [2, 5, 10^{-7}], [2, 3, 10^{-7}], [2, 5, 10^{-7}], [2, 6.5, 10^{-7}], [2, 10, 10^{-7}], [2, 5.5, 10^{-6}], [2, 5.5, 10^{-4}] and [2, 5.5, 5×10^{-6}], the ST controller cannot have valid performance which gives zero queue length.



(a) **(b)**

Fig. E.1: Instantaneous Queue Size of ST Controller using Different W_s parameters values

Fig E.1 shows more experiments when using different W_s parameters values than the determined ST controllers in Section 4.3.1.1. For more clarity, Fig. E.1a compares experiments 1 and 3 to the determined ST controller and Fig. E.1b, compares experiments 3 and 4 to the determined ST controller. One can see that the queue size for experiment 2 when the parameter M=4 is not able to perform stable performance all the time since it goes to zero at t=130sec and 180 sec. As this figure shows, the queue size for the experiment 2 is able to reach to the desired queue size in longer rise/fall time T_r (~21 sec for the experiment 2 to 9sec for the determined ST controller). Moreover, the queue sizes for experiments 3 ad 4 have larger peak queue value than the determined ST controller (~375 and 370 for the experiment 3 and 4 respectively to 301 for the determined ST controller). Therefore, the determined ST controller which uses W_s parameters values as determined in Section 4.2.2.1, performed better.

Appendix F
Effect of the Design Parameters on the STU controller

This appendix investigates the effects of weight functions W_S, W_U and W_T on the system performance using the STU controller in Section 4.4. Particularly, we vary the parameters of W_T while fixing W_S and W_U. Then, we shall vary the weight functions W_S, W_U while W_T is fixed.

F.1 Effect of Varying W_T

We shall investigate the effect of varying W_T while W_S and W_U are fixed. We shall use $W_S(s) = \frac{s+11}{2s+11\times10^{-7}}$ and $W_U(s) = 10^5$ in this study which have been determined in Section 4.2 for the SU controller, and vary the parameters in $W_T(s) = \frac{a_1+a_2 s}{1+a_3 s}$.

a) Varying a_1

In order to explore the impact of a_1, we set a_1 to different values but fixing a_2 and a_3 at 3.5 and 0.01 respectively.

Table F.1: Different STU Controllers Specifications by Varying a_1

a_2=3.5, a_3=10^{-2}	γ_{min}	$\|E\|_\infty$	$\|E_2\|_2$	$e(\infty)$	PM, GM
a_1=0.00031	30.0838	21.7674	20.7658	2.0451e-006	66.08 °, 19.01 dB
a_1=0.001	30.0848	21.6789	20.8595	2.0073e-006	65.99 °, 18.96 dB
a_1=0.01	30.0842	21.6821	20.8554	2.00316e-006	65.99 °, 18.96 dB
a_1=1	30.0990	21.7250	20.8321	1.9828e-006	66.01 °, 18.96 dB
a_1=2.5	30.1671	21.8351	20.8154	2.06740e-006	66.07 °, 19.00 dB

Table F.1 compares different controllers specifications including γ_{min}, $\|E_\infty\|_\infty$, $\|E_2\|_2$, $e(\infty)$, the phase margin PM, and the gain margin GM as a function of a_1 while fixing a_2, a_3 and $W_S(s)$, $W_U(s)$. By increasing a_1, one can see that $\|E_\infty\|_\infty$, $\|E_2\|_2$ and γ_{min} all increase and give rise to a less robust controller but none of these differences are statistically significant. In addition, no significant differences are found between $e(\infty)$, PM and GM values by increasing

a_1. Table F.1 shows that all the controllers are stable regarding to positive PM and GM.

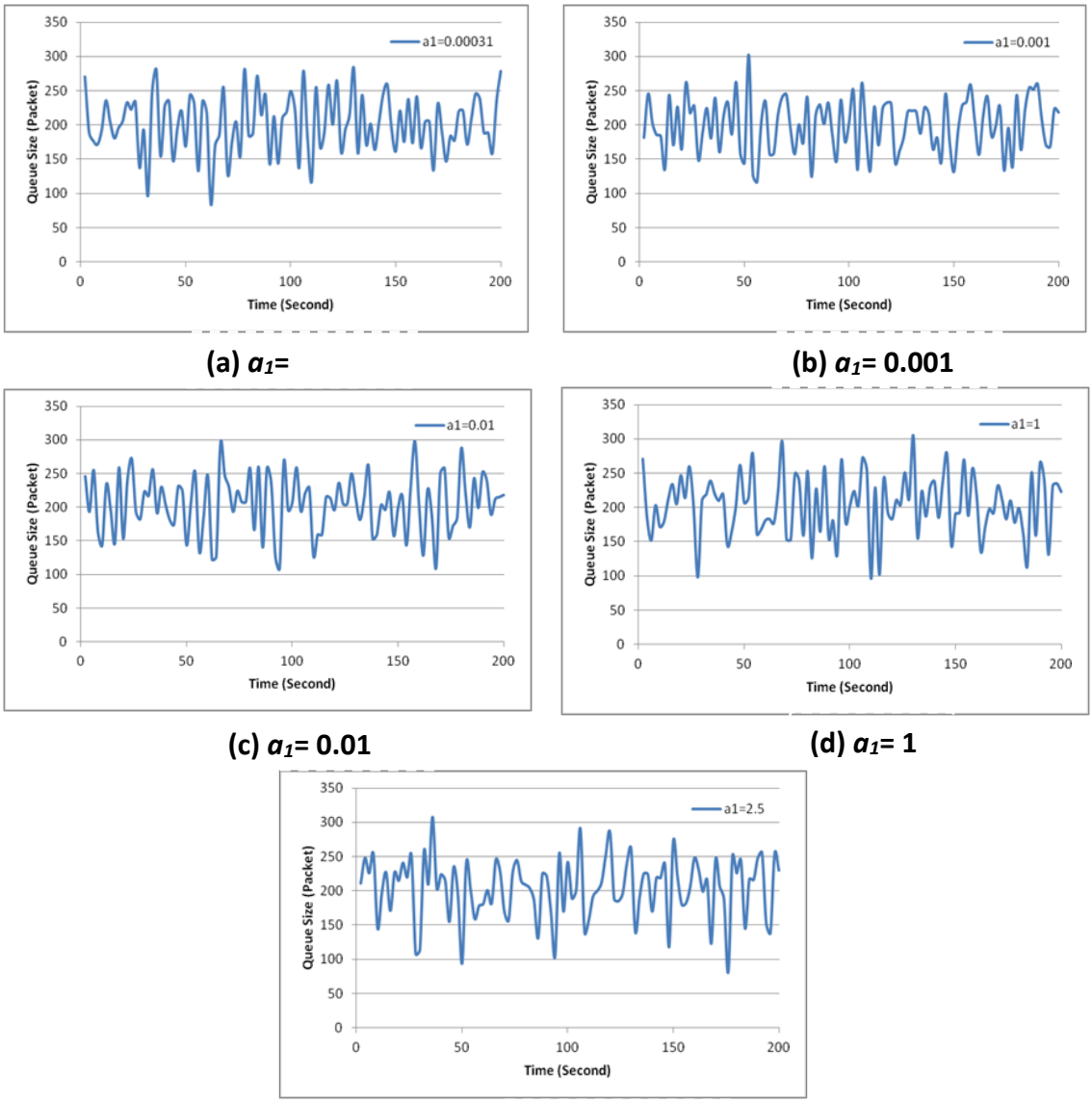


Fig. F.1: Instantaneous Queue Size of STU Controllers with a_2=3.5 a_3=0.01 But Different a_1 Values

The instantaneous queue size for the *STU* controller simulated in OPNET are presented in Fig F.1. It is apparent from the Fig F.1 that a_1= 2.5 gives longest T_r and the largest queue size peak value. Data from this figure are extracted to Table F.2 and it compares the T_r, peak value, μ_q, σ_q as a function of a_1. One can see that the T_r, peak value, μ_q, σ_q are not monotonically increasing w.r.t. a_1. The best results with the smallest T_r, μ_q and σ_q are achieved for a_1=0.001. In this case,

it has the smallest standard deviation hence less deviation from the mean. Therefore we choose a_1=0.001 for our final STU controllers.

Table F.2: Performance of STU Controllers with Different a_1

a_2=3.5, a_3=10^{-2}	T_r (Second)	Peak Value (Packet)	μ_q (Packet)	σ_q(Packet)
a_1=0.00031	3	271	201	42
a_1=0.001	1	250	200	39
a_1=0.01	2	245	204	43
a_1=1	2	271	204	44
a_1=2.5	8	248	204	44

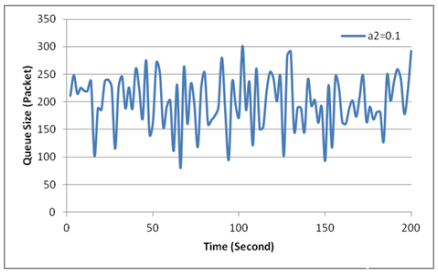
b) Varying a_2

In this section, we investigate the impact of a_2 by setting a_2 to different values but fixing a_1 and a_3 at 0.001 and 0.01. In addition, W_S, W_U are fixed at $W_s = \frac{s+5.5}{2s+11\times10^{-7}}$ and $W_U = 10^7$. Different STU controllers are derived corresponding to different values of a_2.

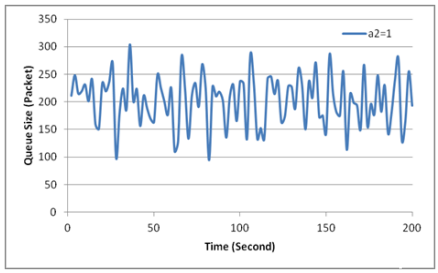
Table F.3 presents the summary statistics controllers specifications including γ_{min}, $\|E_\infty\|_\infty$, $\|E_2\|_2$, $e(\infty)$, the phase margin PM, and the gain margin GM as a function of a_2 while fixing a_1, a_3 and $W_S(s)$, $W_U(s)$. A clear benefit of increasing a_2 in γ_{min}, $\|E_\infty\|_\infty$, $\|E_2\|_2$, $e(\infty)$, PM and GM cannot be seen. All the controllers are robust regarding to have finite γ_{min} values and are stable regarding to positive PM and GM.

Table F.3: Different STU Controllers Specifications by Varying a_2

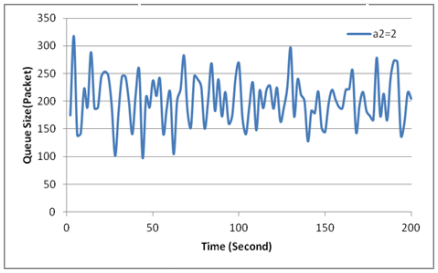
a_1=0.001, a_3=10^{-2}	γ_{min}	$\|E_\infty\|_\infty$	$\|E_2\|_2$	$e(\infty)$	PM, GM
a_2=0.1	30.0892	21.6447	20.9014	2.0450e-006	65.98°, 18.96 dB
a_2=1	30.0758	21.6623	20.8638	2.0529e-006	18.98°, 66.04 dB
a_2=2	30.0788	21.7123	20.8160	2.0220e-006	66.02°, 18.98 dB
a_2=2.5	30.0844	21.6898	20.84756	2.0468e-006	66.03°, 18.99 dB
a_2=3.5	30.0848	21.6789	20.8595	2.0073e-006	65.99°, 18.96 dB
a_2=4.5	30.0832	21.7069	20.8282	1.8750e-006	66.00°, 18.91 dB
a_2=6	30.0951	21.7344	20.8166	2.0029e-006	66.05°,18.98 dB
a_2=7	30.1050	21.7326	20.8327	1.992e-006	66.04°, 18.98 dB
a_2=8	30.1067	21.7483	20.8188	2.0706e-006	66.20°, 19.03 dB



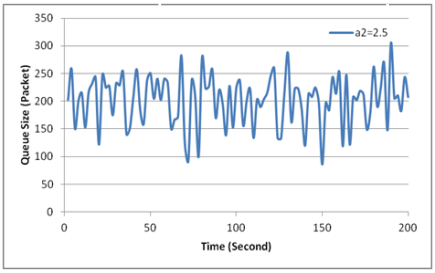
(a) a_2=0.1



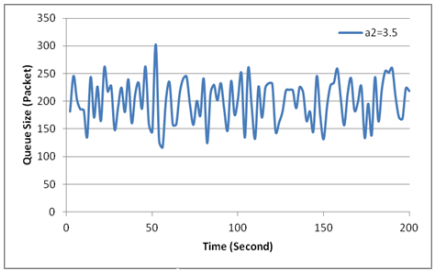
(b) a_2=1



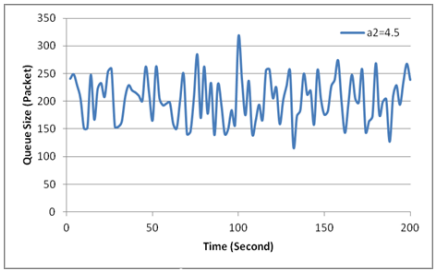
(c) a_2=2



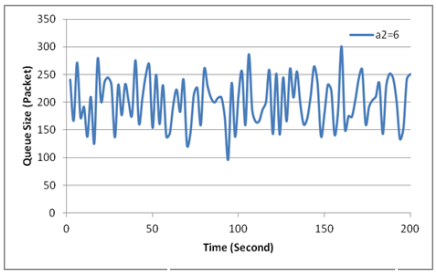
(d) a_2=2.5



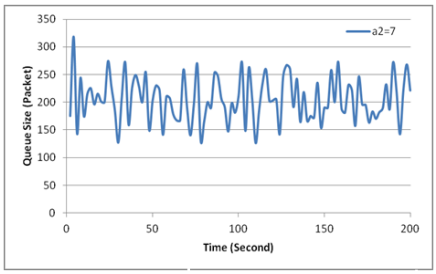
(e) a_2=3.5



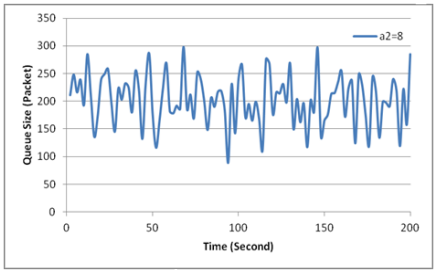
(f) a_2=4.5



(g) a_2=6



(h) a_2=7



(i) a_2=8

Fig. F.2: Instantaneous Queue Size of STU Controllers with a_1=0.001, a_3=0.01 but Different a_2 Values

Table F.4: Performance of STU Controllers with Different a_2

a_1=0.001, a_3 =10^{-2}	T_r (Second)	Peak Value (Packet)	μ_q (Packet)	σ_q(Packet)
a_2=0.1	14	251	200	49
a_2=1	14	246	202	45
a_2=2	4	316	202	43
a_2=2.5	5	255	202	45
a_2=3.5	6	245	200	39
a_2=4.5	7	249	204	41
a_2=6	2	243	202	44
a_2=7	4	318	204	40
a_2=8	9	248	202	45

Fig F.2 presents the instantaneous queue size for a_2 ranging from 0.1 to 8 while other parameters are fixed. From Fig. F.2, it is apparent that all the controllers are stable and are able to reach to desired queue size, 200. However, it can be seen that there is not a clear trend of T_r or peak value by increasing a_2. In order to have further look at data of different controllers simulation, Table F.4 compares the summary statistics for different STU controllers by varying a_2. From the table, one can see that there is not a clear trend, and not significant differences between μ_q and σ_q by increasing a_2. In addition, except a_2=2 and 7, peak value of queue size stays in the same range by increasing a_2. It is apparent from the Table F.4 that a_2=6 gives the smallest T_r and peak queue size value. Hence, a_2=6 is chosen for final STU controller.

c) Varying a_3

In this subsection, a_3 is set to different values, while a_1 and a_2 are kept unchanged at 0.001 and 6 respectively. As mentioned earlier, we will use $W_S(s)$, $W_U(s)$ chosen in SU controller which $W_s = \frac{s+11}{2s+11\times10^{-7}}$ and $W_U = 10^7$.

Table F.5: Different STU Controllers Specifications by Varying a_3

a_1=0.001, a_2=6	γ_{min}	$\|E_\infty\|_\infty$	$\|E_2\|_2$	$e(\infty)$	PM, GM
a_3=0.0001	30.3142	23.1856	19.5289	2.1686e-006	66.54 °, 19.55 dB
a_3=0.001	30.0993	21.8637	20.6867	1.8866 e-006	66.07 °,18.98 dB
a_3=0.01	30.1067	21.7483	20.8188	2.0706e-006	66.20 °, 19.03 dB
a_3=0.05	30.0901	21.6916	20.8540	2.0604e-006	66.14 °, 19.00 dB
a_3=0.1	30.0889	21.7025	20.8409	2.0578e-006	66.15 °,19.00 dB
a_3=0.5	30.0838	21.6834	20.8533	2.0509e-006	66.11 °, 18.97 dB

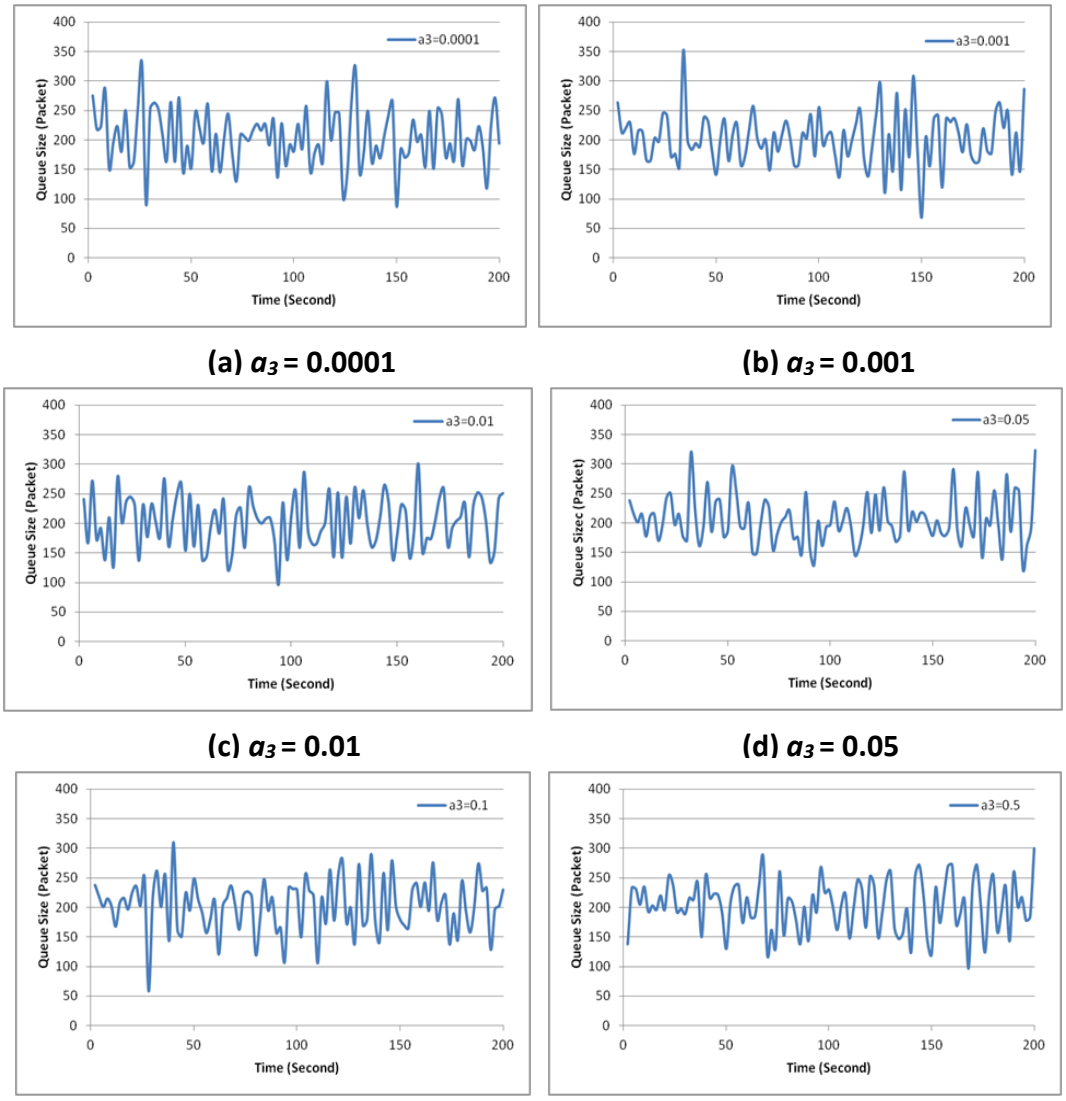
Table F.5 provides different controllers specifications including γ_{min}, $\|E\|_\infty$, $\|E_2\|_2$, $e(\infty)$, PM and GM as a function of a_3. Similar to Tables F.3 and F.4, no significant differences are found between different controllers. All the controllers are robust due to finite γ_{min} and stable regarding to positive PM and GM.

Fig F.3 presents the instantaneous queue size for a_3 ranging from 0.0001 to 0.1 while other parameters are fixed. One can see that all the controllers are stable and robust since the results reaches to desired queue size, 200. In order to have deeper look at OPNET simulation results, Table F.6 shows the summary statistics for different STU controllers by varying a_3. It shows that peak queue size value is decreasing w.r.t. a_3 but there is no other trend in the table. It is apparent from the Table F.6 that a_3=0.01 has the shortest T_r while has reasonable peak value, μ_q and σ_q. Hence, a_3=0.01 is chosen for final STU controller.

According to discussion given above, $W_T(s)$ will be chosen as $W_T(s) = \frac{0.001+6s}{1+0.01s}$.

Table F.6: Performance of STU Controllers with different a_3

a_2=0.001, a_2=6	T_r (Second)	Peak Value (Packet)	μ_q (Packet)	σ_q(Packet)
a_3=0.0001	8	274	203	47
a_3=0.001	8	260	201	44
a_3=0.01	2	242	202	44
a_3=0.05	8	237	204	40
a_3=0.1	9	237	203	44
a_3=0.5	10	233	203	42



(a) a_3 = 0.0001 **(b) a_3 = 0.001**

(c) a_3 = 0.01 **(d) a_3 = 0.05**

(e) a_3 = 0.1 **(f) a_3 = 0.5**

Fig. F.3: Instantaneous Queue Size of STU Controllers with a_1=0.001, a_2=6 but Different a_3 Values

F. 2 Effect of Varying W_u and W_s

In this section, we shall investigate the effect of varying W_u and W_s while W_T is fixed. Specifically, we vary the parameters ω_B, M or A *in* W_S one in time while fixing W_T and W_u. Then, we vary the W_u while fixing W_s and W_T. Note that we shall use the same example as in Section 4.2.2.1.

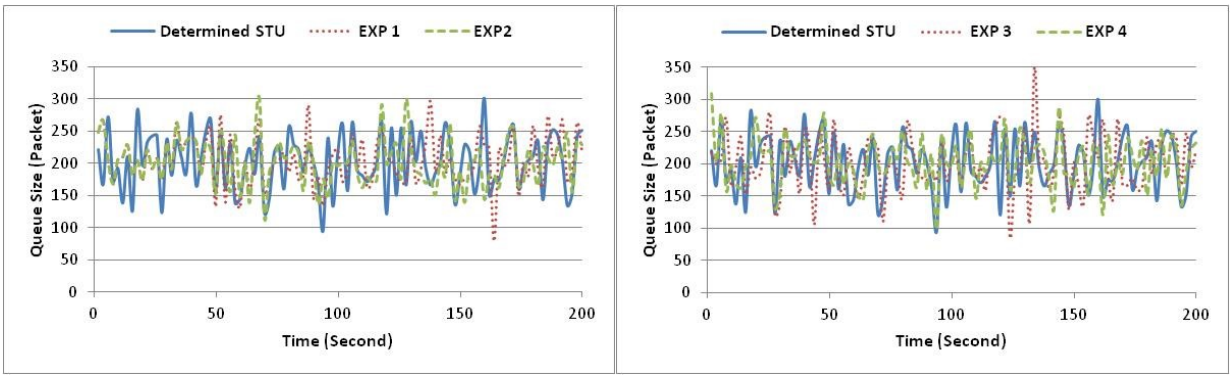
Table F. 7 shows different experiments when W_u or W_s is varying. We vary different W_s parameters value one at a time. We have experimented several values for different varying

parameters. However, for avoiding repetition and lack of space, we only put two best cases for each varying parameters. In experiments 1 and 2, the parameters M is varying when ω_B, A, W_u and W_T are fixed. In experiments 3 and 4, the parameters value of ω_B is varying while other parameters are fixed. In experiments 5 and 6, we have varying A scenario while ω_B, M, W_u and W_T are fixed. In experiments 7 and 8, W_u is varying while W_T and W_s is fixed.

Table F.7: Different Weight Functions for More Experiments for the STU controller

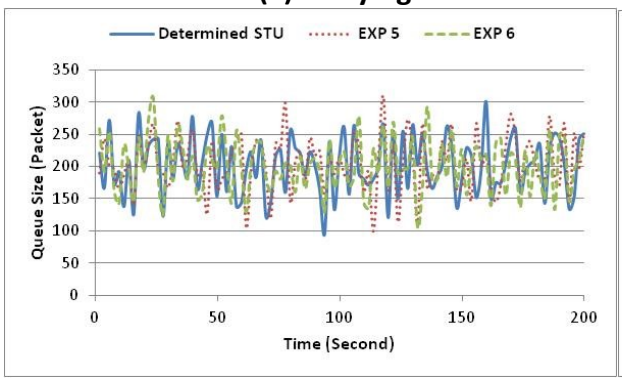
$W_T(s) = \frac{0.001 + 6s}{1 + 0.01s}$	$W_s(s) = \frac{s/M + \omega_b}{s + \omega_b \times A}$			W_u
	M	ω_b	A	
Determined STU	4	5.5	10^{-7}	10^5
EXP 1	6	5.5	10^{-7}	10^5
EXP 2	2	6.5	10^{-7}	10^5
EXP 3	2	5	10^{-7}	10^5
EXP 4	2	5.5	10^{-7}	10^5
EXP 5	2	5.5	10^{-6}	10^5
EXP 6	2	5.5	10^{-5}	10^5
EXP 7	2	5.5	10^{-7}	10^6
EXP 8	2	5.5	10^{-7}	5×10^5

Fig. F.4 compares the performance of the STU controller when W_u or W_s is varying to the determined STU controller in Section F.1. One can see that in Fig. F.4.a, the determined STU controller has smaller rise/fall time T_r to experiments 1 and 2 (~2 sec for the determined STU controller to 4 sec and 5 sec for the experiments 1 and 2 respectively). Fig. F.4.b compares the performance of the STU controller while ω_B is varying. One can see that the determined controller has smaller rise/fall time T_r to the experiments 3 and 4 (~2 sec for the determined STU controller to 3 sec and 4 sec for the experiments 3 and 4 respectively). In addition, the determined STU controller has smaller peak queue value to the experiment 4 (~242 packets for

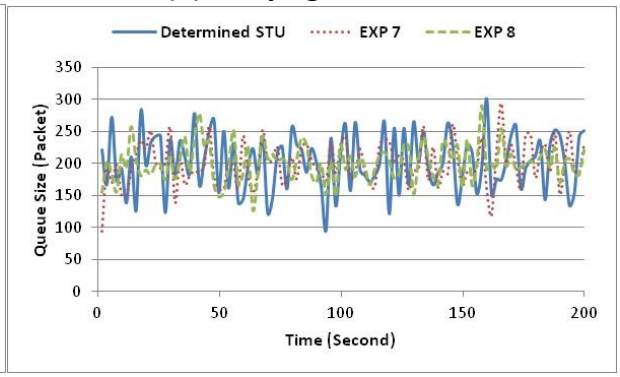


(a): Varying *M*

(b): Varying ω_B



(c): Varying *A*



(d): Varying W_u

Fig. F.4: Instantaneous Queue Size of STU Controller using Different W_s and W_u parameters values

the determined STU controller to 350 packets for the experiment 4). Fig. F.4.c compares the performance of the STU controller while *A* is varying. One can see that the determined controller has smaller rise/fall time T_r and peak queue value to the experiments 5 and 6 (~2 sec and 242 packets for the determined STU controller to 3 sec and 301 packets and 4 sec and 303 packets for the experiments 5 and 6 respectively). Fig. F.4.d compares the performance of the STU controller while W_u is varying. One can see that the determined controller has smaller rise/fall time T_r to the experiments 7 and 8 (~2 sec for the determined STU controller to 16 sec and 10 sec for the experiments 7 and 8 respectively). Therefore, the determined STU controller shows better performance than other scenarios shown in Table F.7.

Appendix G
RB-SU Example on Weight Function

In order to investigate the feasibility of our weight functions selected for $W_s(s)$ and $W_u(s)$ for the RB-SU controller in Section 5.3, similar to Section 5.2.1, the bode diagram of the sensitivity functions and their associated weight functions shall be shown in this Appendix.

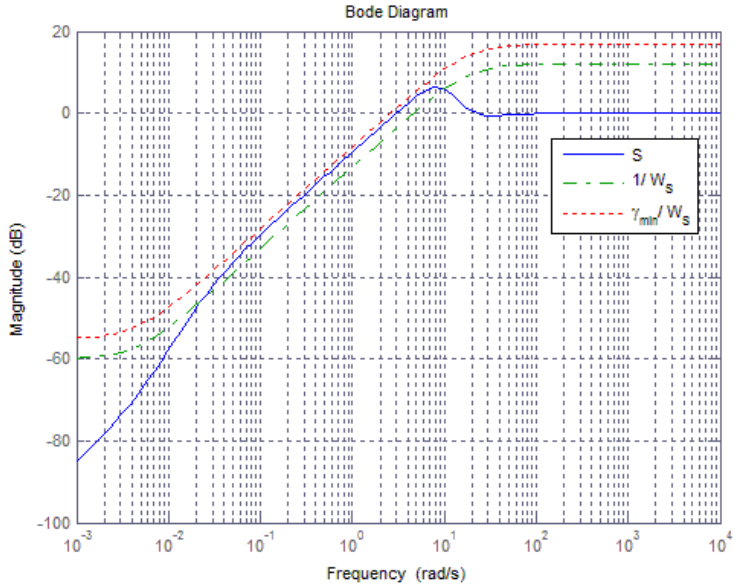


Fig G.1: Diagram of S, $1/W_S$ and γ_{min}/W_S for the *RB-SU* Controller

In order to investigate the feasibility of our weight functions selected for $W_s(s)$, Fig G.1 presents the Bode diagrams of the sensitivity function $S(s)$ (specified by the solid blue curve), the inverse of the sensitivity weight function $1/W_S$ (specified by dotted red curve) and γ_{min}/W_S (specified by dashed green curve. The sensitivity of the system output to disturbance and the error between system output and the reference input is small at low frequencies due to small sensitivity function, $S(s)$ at low frequencies. Contrarily, the magnitude of $S(s)$ attains the asymptotic value 1 (0dB) at high frequencies which demonstrate that the noise and influence on the tracking error would be attenuated [DoBi10]. Moreover, Fig G.1 has verified that $|S(j\omega)|$

is upper bounded by $\gamma_{min}/|W_S|$ which is one of the requirements in designing SU controllers, please see section 3.4 for more details.

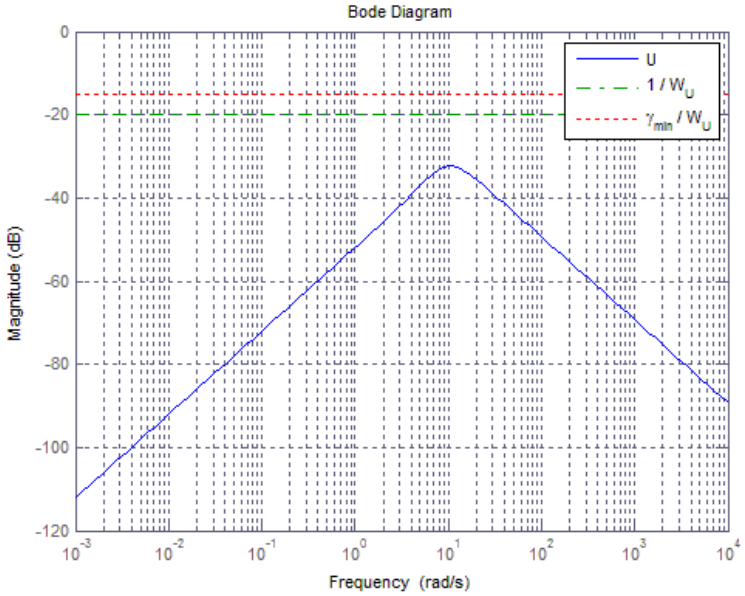


Fig. G.2: Bode Diagram of U, $1/W_U$ and γ_{min}/W_U for the RB-SU Controller

Comparably, Fig. G.2 represents the Bode diagrams of the input sensitivity function, *U(s)*, the inverse of the sensitivity weight function $1/W_U$ and γ_{min}/W_U. This figure investigates that an appropriate $W_u(s)$ has been selected while $|U(j\omega)|$ is upper bounded by $\gamma_{min}/|W_U$.

Appendix H
More Experiments to Recommend Weights Parameters for the RB-SU controller

We shall provide more experiments here to show how to recommend weight function parameters for the RB-SU controller in Section 5.3. Specifically, we provide here deeper insight on the selection of M and ω_B.

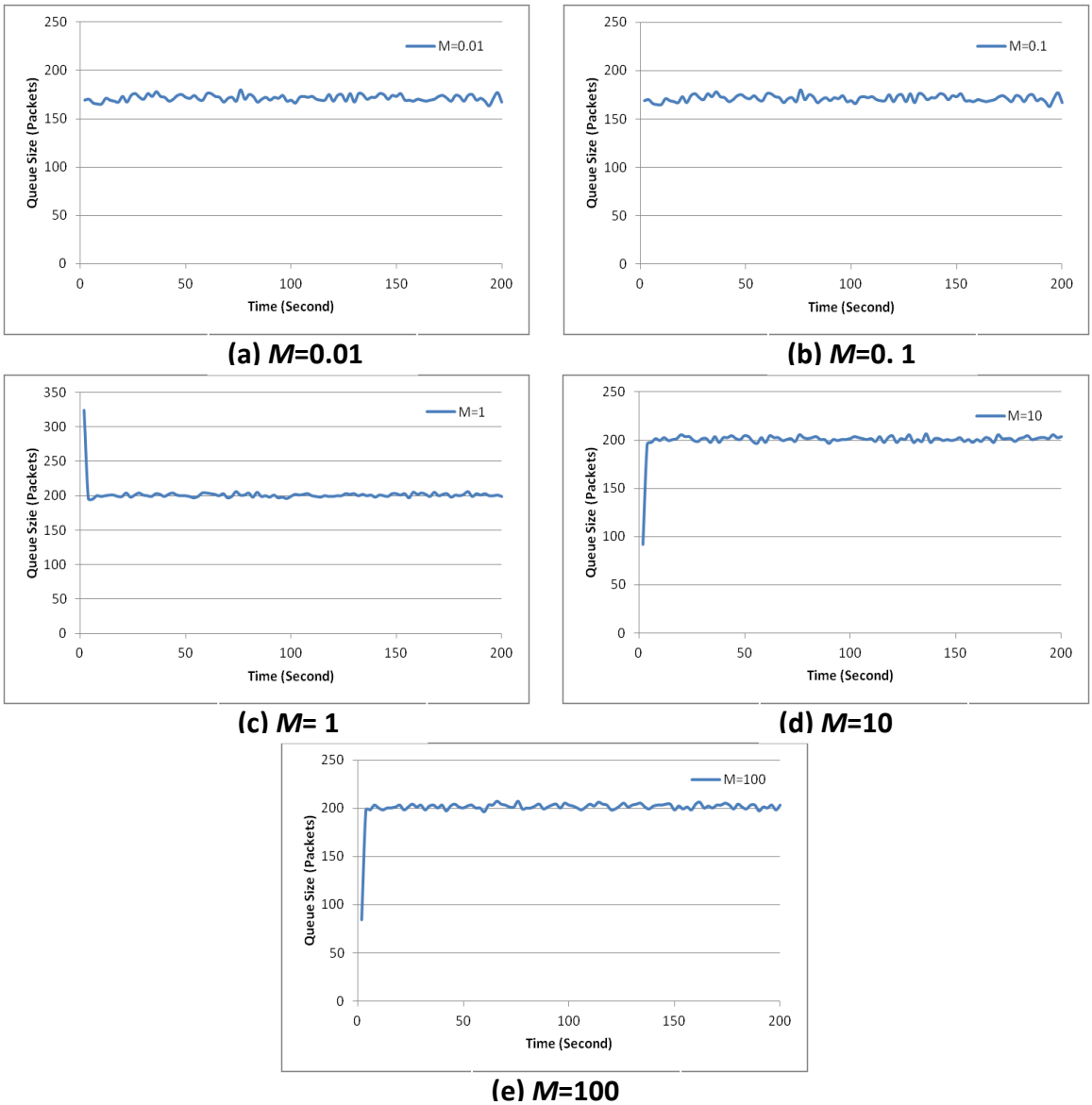
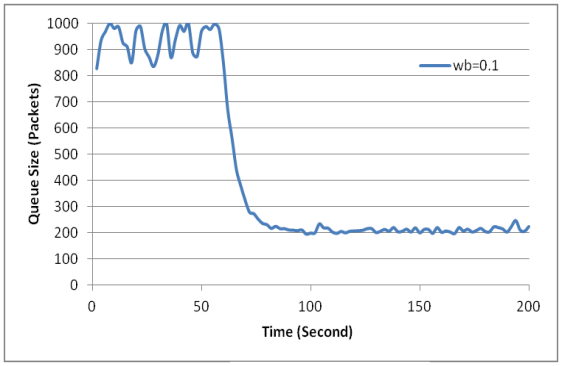


Fig. H.1: Instantaneous Queue Size of the RB-SU Controllers with W_U=10, ω_B= 4.5 and A= 10^{-3} but Different M Values

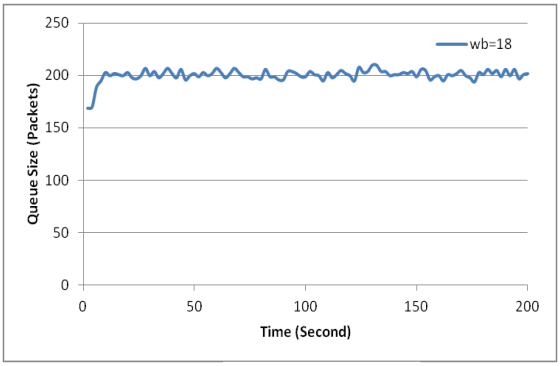
H.1 Parameter *M*

We shall investigate different range of *M* to find suitable range of it for the RB-SU controller. In this section, we vary *M* from *M*=0.1 to 100 while fixing W_U=10, ω_B= 4.5 and *A*= 10^{-3}.

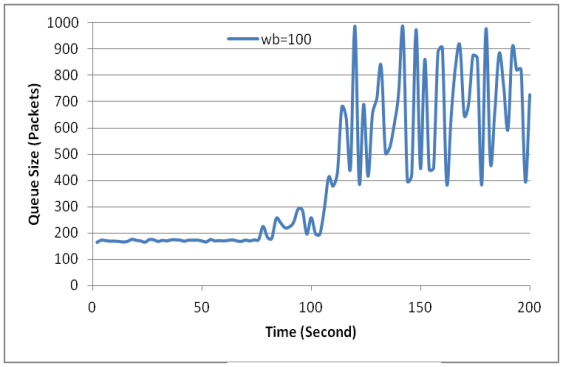
Fig H.1 shows the instantaneous queue size for the RB-SU controller simulated in OPNET for wide range of *M* from *M*=0.1 to 100 while fixing W_U=10, ω_B= 4.5 and *A*= 10^{-3}. One can see from Fig 5.6 and Fig H.1 that for *M* greater than 1, the reasonable instantaneous queue size with small T_r and stable response which can reach to the desired queue size value is achievable. Also, the show that by increasing the *M* greater than 5, queue size does not change significantly. Moreover, Fig H.1.a-b shows that the queue size cannot reach to the desired queue size value for *M* less than 1. Therefore, the *M* is set to be larger than 1. One can also see that there are not significant changes by increasing *M*. Therefore, in determining the *M*, we need to choose *M* be greater than 1 in order to have reasonable performance like reach to desired value of queue size, acceptable rise/fall time.



(a) ω_B =0.1



(b) ω_B =18



(c) ω_B =100

Fig. H.2: Instantaneous Queue Size of the RB-SU Controller with W_U=10, *M* = 5 and *A*= 10^{-3} but Different ω_B Values

H.2 Parameter ω_B

We shall provide more simulation results while ω_B is varying and other parameters are fixed to state clear the recommendation comments clearly in Section 5.3.2.1.4. In this section, we vary ω_B in a wide range from ω_B =0.1 to 100 while fixing W_U=10, ω_B= 4.5 and A= 10^{-3}.

Fig. H.2 shows the instantaneous queue size for the *RB-SU* controller simulated in OPNET for wide range of ω_B from ω_B =0.1 to 100 while fixing W_U=10, ω_B= 4.5 and A= 10^{-3}. It illustrates that for small ω_B=0.1, the queue size begins from large queue size value before going to stable state at t=95 sec. Moreover, for very large ω_B=100, the queue size is unstable. For ω_B=18, the queue size is reasonable; It reaches to the desired queue size value at t=10 sec with small variation.

Appendix I
Effect of the Design Parameters on the RB-STU Controller

Similar to the RB-SU controller in section 5.3.2, simulations have been executed to assess the effect of the parameters on the performance and robustness of the RB-STU controller. Specifically, the parameters values of the $W_T(s)$, different values of a_1, a_2, a_3, are investigated in the interest of observing their impacts on the system performance while fixing W_S and W_u at the value determined at Section 5.3.2. In order to show the effectiveness of our determined parameters values, we shall investigate the performance of the controller by having different W_s and W_u parameters values while W_T is fixed. We also shall provide more experiments on wider range of W_T parameters to have deeper insight for recommending W_T parameters selection in Chapter 7.

I.1 Effect of Varying W_T

We shall fix at $W_s(s) = \frac{s/5+4.5}{s+4.5\times10^{-3}}$ and W_u=10 in this study and vary the parameters in $W_T(s) = \frac{a_1+a_2 s}{1+a_3 s}$. Also, see Section 3.2 for the parameters definition.

a) Varying a_1

To find the best a_1, we vary a_1 while keeping fix W_s, W_u, a_2 and a_3 at $W_s(s) = \frac{s/5+4.5}{s+4.5\times10^{-3}}$, W_u=10, a_2=1 and a_3=0.1.

Table I.1: Different RB-STU Controllers Specifications by Varying a_1

a_2=1, a_3=0.1	γ_{min}	$\|E_\infty\|_\infty$	$\|E_2\|_2$	$e(\infty)$	PM, GM
a_1=0.001	2.7071	2.6537	0.5352	1.17 e-13	71.7°, 12 dB
a_1=0.002	2.7072	2.6537	0.5352	3.56 e-14	71.7°, 12 dB
a_1=0.01	2.7072	2.6538	0.5353	1.86 e-14	71.7°, 12 dB
a_1=0.05	2.7075	2.6542	0.5347	7.32 e-15	71.7°, 12 dB
a_1=0.1	2.7084	2.6550	0.5350	2.70 e-14	71.7°, 12 dB

Table I.1 shows different controller specifications including γ_{min} (the minimum mixed H_2 /H_∞-norm), $\|E_\infty\|_\infty$ (the H_∞-norm of closed-loop transfer functions from ***w*** to ***z***), $\|E_2\|_2$ (the H_2-norm of closed-loop transfer functions from ***w*** to ***z***,), the steady state error $e(\infty)$, the phase margin PM, and the gain margin GM as a function of a_1. By increasing a_1, one can see that $\|E_\infty\|_\infty$, $\|E\|_2$ and their corresponding γ_{min} increase (though not significantly). The controllers are stable due to their positive phase margin and gain margin.

Fig. I.1 shows the instantaneous queue size for the RB-STU controller when simulated in OPNET for a_1 ranging from a_1=0.001 to 0.1. One can see that all queue sizes are relatively similar. They all reach steady state in a short time and have remarkably small variations. In order to have a deeper insight, Table I.2 provides different performance measures including T_r (rise/fall time), maximum and minimum packet size value, μ_q (mean queue length) and σ_q (queue length standard deviation). One can see that, T_r and μ_q are increasing w.r.t. increasing a_1. The smallest T_r (~5 Sec) belongs to a_1= 0.001, 0.002 and 0.1 and the smallest μ_q (~211 packets) belongs to a_1= 0.001. One can see that max/min, σ_q are not changing monotonically w.r.t. a_1. The smallest σ_q (i.e. the least variation) and peak queue value (700 packets) belongs to a_1= 0.001 and. Therefore, a_1= 0.001 is chosen for its smallest values for all the measurements.

Table I.2: Performance of RB-*STU* Controllers with Different a_1

a_2=1, a_3=0.1	T_r (Sec)	Max/Min Value (Packet)	μ_q (Packet)	σ_q(Packet)
a_1=0.001	5	700/197	211	63
a_1=0.002	5	993/198	215	89
a_1=0.01	5	951/197	215	86
a_1=0.05	6	935/197	216	100
a_1=0.1	6	999/197	216	98

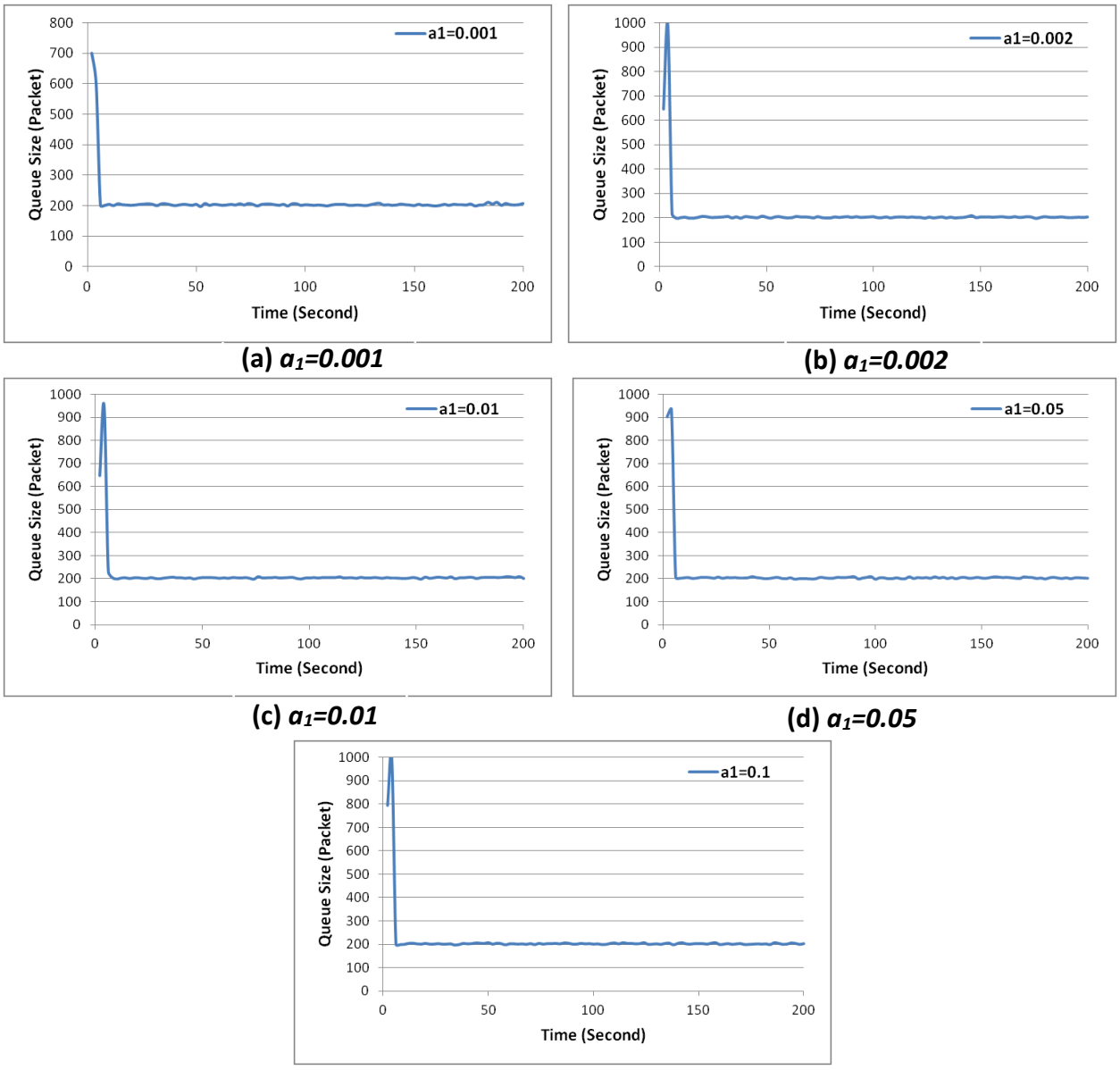


Fig. I.1: Instantaneous Queue Size of the RB-*STU* Controllers with a_2=1, a_3=0.1 but Different a_1 Values

b) Varying a_2

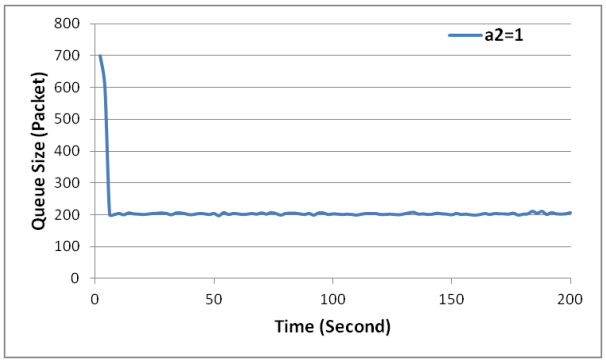
In this section, we observe the effect of a_2 by setting a_2 to different values while fixing W_s, W_u, a_1 and a_3 at $W_s(s) = \frac{s/5+4.5}{s+4.5\times10^{-3}}$, W_u=10, 0.001 and 0.1 respectively.

Table I.3: Different RB-STU Controllers Specifications by Varying a_2

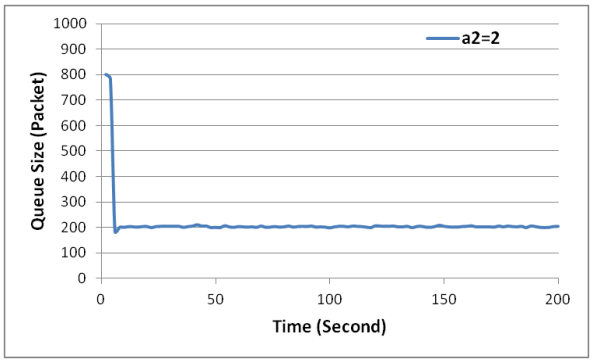
a_1=0.001, a_3=0.1	γ_{min}	$\|E_\infty\|_\infty$	$\|E_2\|_2$	$e(\infty)$	PM, GM
a_2=1	2.7071	2.6537	0.5352	1.17 e-13	71.7°, 12 dB
a_2=2	3.5275	3.4904	0.5102	8.21 e-15	76.38 °, 15dB
a_2=2.75	4.0248	3.9926	0.5082	1.91 e-13	78.2 °, 16dB
a_2=3	4.1759	4.1450	0.5069	9.57 e-14	78.6°, 16dB
a_2=3.25	4.3213	4.2918	0.5042	3.04e-14	79.0°, 17dB
a_2=3.5	4.4609	4.4323	0.5045	4.19e-14	79.4°, 17dB
a_2=4	4.7276	4.7001	0.5089	2.75e-14	80.1°, 18dB
a_2=5	5.2171	5.1916	0.5145	1.78e-14	81.1°, 19dB

Table I.3 shows different designed controllers specifications including γ_{min}, $\|E_\infty\|_\infty$, $\|E_2\|_2$, $e(\infty)$, PM and GM. By increasing a_2, $\|E_\infty\|_\infty$ and γ_{min} are increasing but not significantly. There is no monotonic increase in steady state error and $\|E_2\|_2$. The largest error and $\|E_2\|_2$ occur at a_3=2.75 and a_3=1 respectively and the smallest at a_3=2 and a_3=3.25 respectively. All controllers are stable due to their positive PM and GM values.

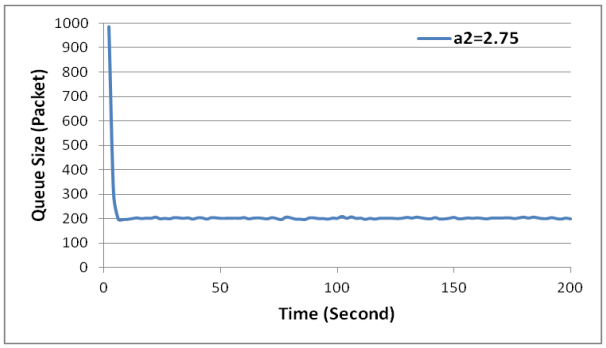
Fig. I.2 illustrates the instantaneous queue size for a_2 ranging from a_2=1 to 5. One can see that they all reach steady state in a short time and have small variations. For the controller designed by smaller than a_2=3, unlike other controllers, queue sizes start from relatively larger value than 200. In order to have a deeper insight, Table I.4 provides different performance measures including T_r (rise/fall time), maximum and minimum packet size value, μ_q (mean queue length) and σ_q (queue length standard deviation). One can see that, T_r, max/min, μ_q, σ_q are not changing monotonically w.r.t. a_2. The smallest T_r (~4s) belongs to a_2=3 while the longest T_r (~20s) belongs to a_2 =4. The smallest peak queue value (~206 packets) belongs to a_2 =3.5 while the smallest μ_q (~202 packets) belongs to a_2 =3. The largest peak queue value (~987 packets) belongs to a_2 =2.75. Moreover, the two smallest σ_q belong to a_2=5 and 20 (~7 packets) and a_2=5 (~9packets). The reason of having large σ_q for a_2 smaller than a_2=4 is relatively large peak queue value at the beginning but not for large oscillations in the steady state. For the final controller, we chose a_2 =3 which gives the smallest T_r (~4s) as well as smallest mean queue with reasonable σ_q and peak queue value.



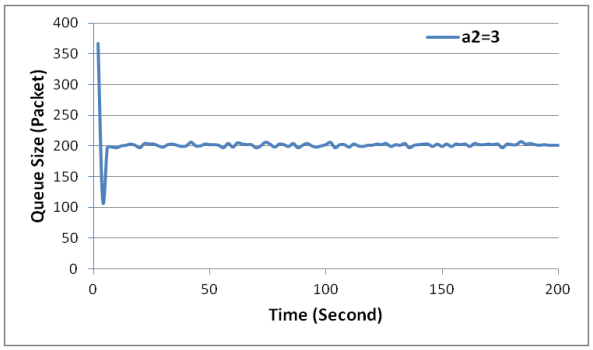
(a) a_2=1



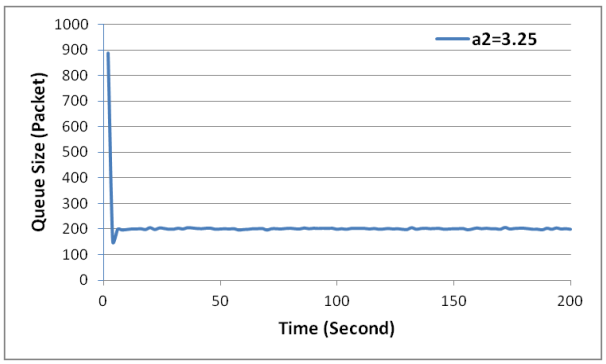
(b) a_2=2



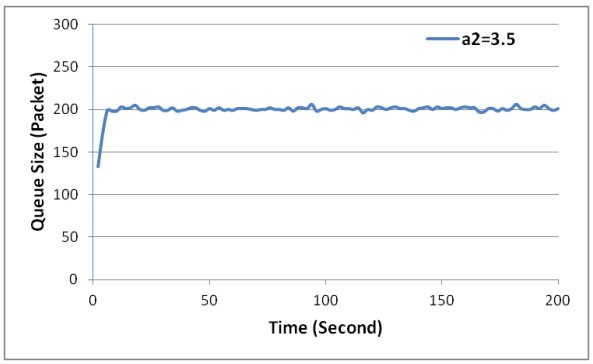
(c) a_2=2.75



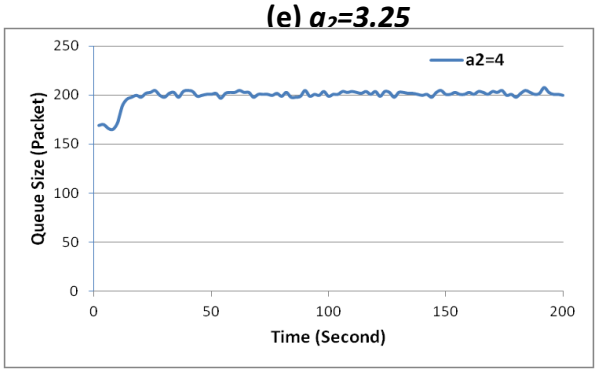
(d) a_2=3



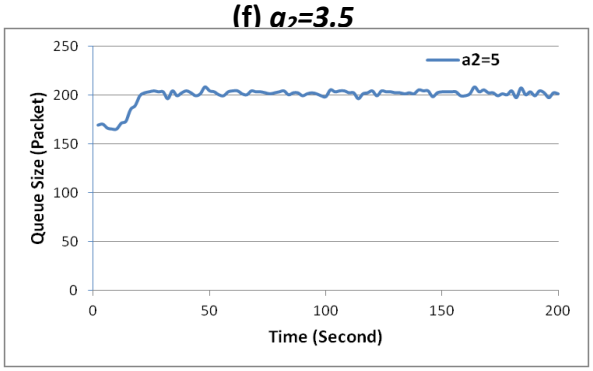
(e) a_2=3.25



(f) a_2=3.5



(g) a_2=4



(h) a_2=5

Fig. I.2: Instantaneous Queue Size of the RB-*STU* Controllers with a_1=0.001, a_3=0.1 but Different a_2 Values

Table I.4: Performance of RB-*STU* Controllers with Different a_2

a_1=0.001, a_3=0.1	T_r (Sec)	Max/Min Value (Packet)	μ_q (Packet)	σ_q(Packet)
a_2=1	5	700/197	211	63
a_2=2	7	801/182	213	82
a_2=2.75	5	987/196	210	78
a_2=3	4	367/111	202	18
a_2=3.25	6	888/151	207	68
a_2=3.5	6	206/133	199	7
a_2=4	20	208/165	199	7
a_2=5	19	208/165	199	9

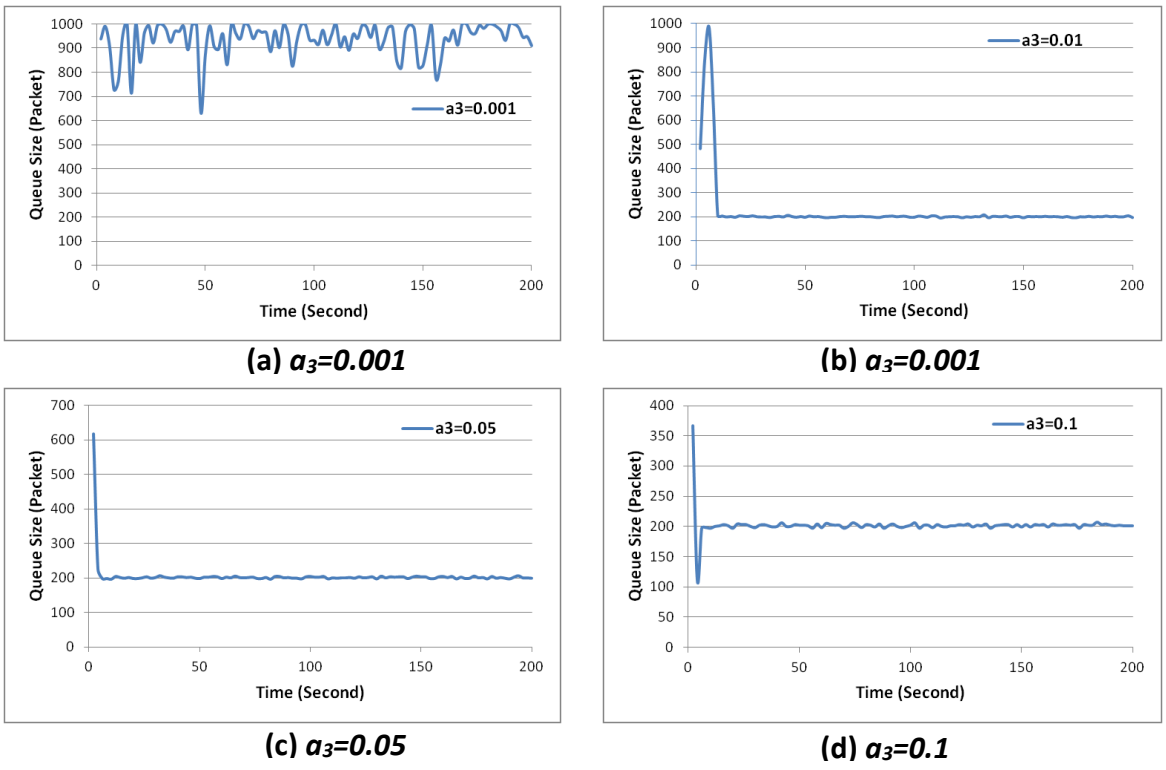
c) Varying a_3

In this section, we observe the effect of a_3 by setting a_3 to different values while fixing W_s, W_u, a_1 and a_2 at $W_s(s) = \frac{s/5+4.5}{s+4.5\times10^{-3}}$, W_u=10, 0.001 and 3 respectively.

Table I.5: Different RB-STU Controllers Specifications by Varying a_3

a_1=0.001, a_2=3	γ_{min}	$\|E_\infty\|_\infty$	$\|E_2\|_2$	$e(\infty)$	PM, GM
a_3=0.001	4.3226	4.3048	0.3919	2.91e-14	76.3°, 17dB
a_3=0.01	4.3100	4.2927	0.3858	2.64e-13	76.5°, 17dB
a_3=0.05	4.2602	4.2371	0.4433	6.50e-14	77.3°, 17dB
a_3=0.1	4.1759	4.1450	0.5069	9.57 e-14	78.6°, 16dB

Table I.5 depicts the specifications of γ_{min}, $\|E_\infty\|_\infty$, $\|E_2\|_2$, $e(\infty)$, PM, and GM as a function of a_3. When a_3 is increasing, $\|E_\infty\|_\infty$, $\|E_2\|_2$ and consequently γ_{min} decrease which results in a more robust controller. Steady state error does not show monotonic changes w.r.t. increasing a_3. The smallest and the largest $e(\infty)$ occurs at a_3 =0.01 and a_3 =0.1 respectively. Furthermore, the positive PM and GM values indicate that the controllers are stable.



(a) a_3=0.001 (b) a_3=0.001

(c) a_3=0.05 (d) a_3=0.1

Fig. I.3 : Instantaneous Queue Size of the RB-*STU* Controllers with a_1=0.001, a_2=3 but Different a_3 Values

Table I.6: Performance of RB-*STU* Controllers with Different a_3

a_1=0.001, a_2=3	T_r (Sec)	Max/Min Value (Packet)	μ_q (Packet)	σ_q(Packet)
a_3=0.001	NA	1000/631	937	70
a_3=0.01	10	987/196	222	112
a_3=0.05	4	617/196	205	41
a_3=0.1	4	367/111	202	18

Fig. I.3 illustrates the instantaneous queue size for a_3 ranging from a_3=0.001 to 0.1. One can see that they all except for a_3=0.001 reach steady state at the desired value (~200 packets) in a short time and have small variations. For the controller designed by a_3=0.001, the system is unstable when the buffer is full. In order to have a deeper insight, Table I.6 provides different performance measures including T_r (rise/fall time), maximum and minimum packet size value, μ_q (mean queue length) and σ_q (queue length standard deviation). One can see that, T_r, max/min, μ_q, σ_q are decreasing w.r.t. a_2. Therefore, the smallest T_r (~4s), least peak queue value (~367 packets), the smallest μ_q (~202 packets) and smallest σ_q (~18 packets) belongs to a_3=0.1.

Therefore, we choose a_3=0.1 for the final controller.

According to the simulation results presented in Sections I.1, we shall choose a_1=0.001, a_2=3 and a_3=0.1 and therefore $W_T(t) = \frac{0.001+3s}{1+0.1s}$ to design the RB-*STU* controller.

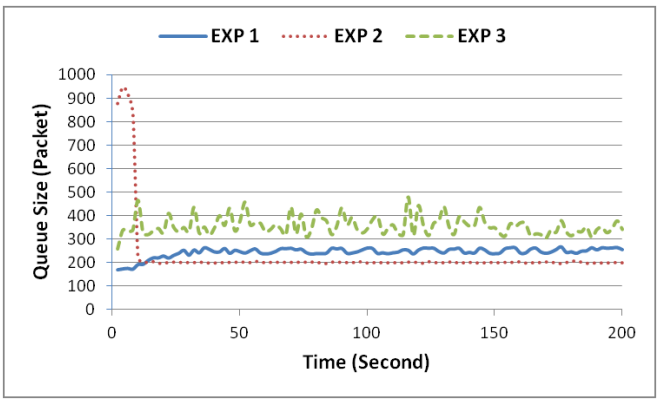
I.2: More Experiments on W_u and W_s

In this section, we shall investigate different W_u and W_s along with the determined W_T in the previous section to show the effectiveness of our designed RB-STU controller. We will compare 5 different experiments with the determined controller in the previous section. In Section I.1, we have used W_u and W_s of the RB-SU controller (determined in Section 5.3) to investigate proper W_T in designing the RB-STU controller.

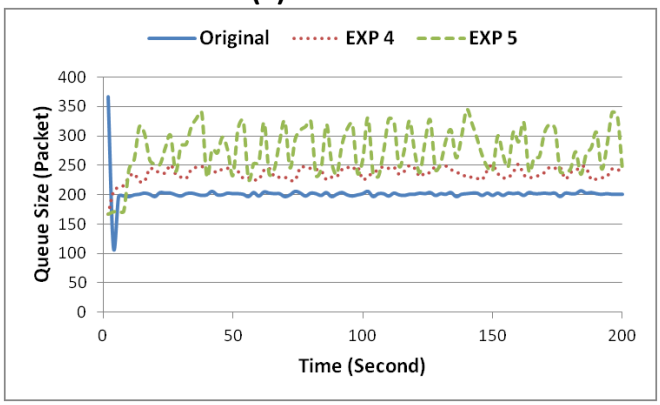
Table I.7: Different Weight Functions for More Experiments of the RB-STU Controller

$W_T(t) = \frac{0.001+3s}{1+0.1s}$	$W_s(s) = \frac{s/M+\omega_b}{s+\omega_b \times A}$			W_u
	M	ω_b	A	
EXP 1	4	6	0.01	10
EXP2	8	3	0.001	10
EXP3	5	4.5	0.01	100
EXP 4	2	8	0.01	100
EXP 5	3	6.5	0.1	10
Original	5	4.5	0.001	10

Table I.7 shows parameters values for the different W_u and W_s used in different experiments. One can see that in all the experiments the W_T is as determined in Section I.1 while varying the parameters for the W_u and W_s. The original values for the W_u and W_s, which are determined in Section 5.3 are also shown on the Table I.7 as well.



(a) *EXP 1-3*



(b) *Original and EXP 4-5*

Fig I. 4: More Experiments for the RB-STU Controller Under Different W_u and W_s

Fig I.4 shows more experiments when using different W_u and W_s parameters than the ones used in Section I.1. For more clarity, Fig. I. 4a shows experiments 1 to 3 and Fig. I. 5, shows experiments 4 to 5 and the original controller. One can see that the queue size for experiments 1, 3, 4 and 5 cannot reach to the desired queue size of 200 packets (~ 235, 360, 235 and 378 packets respectively). Moreover, the queue size for the experiments 3 and 5 show more oscillation than other experiments. The only experiment able to become stabilized around the desired value of 200 packets is experiment 2. However, it has a higher peak value at the beginning than the original one (~938 to 367 packets). These experiments along with other experiments do not show here for limitation on space show the effectiveness of the determined weight functions for the RB-STU controller.

I.3 More Experiments on a_1 and a_2

In this section, we shall provide more simulations results on wider range of W_T parameters

value in order to be able to give W_T weight selection recommendation on Chapter 7.

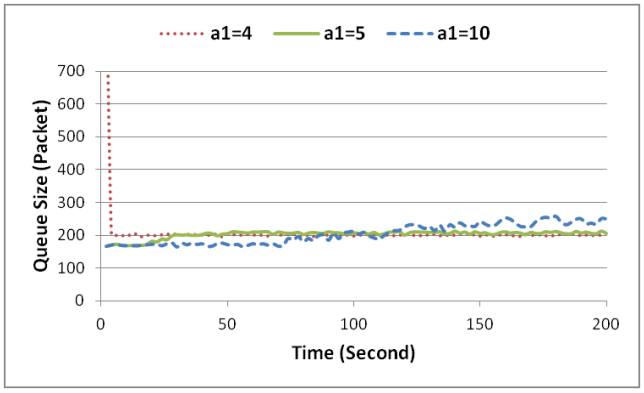


Fig I. 5: More Experiments for the RB-STU Controller Under Wider Range of a_1

Fig I.5 shows more experiments when using wider range of a_1 to be able to give recommendation on W_T weight selection. One can see that for a_1 greater than 4, the controller is not able to perform reasonable. For example, for a_1=5, the queue size reaches to the desired queue size after 30 second which is much longer than the designed controller (~4 second) while for a_1=5, the queue size is not able to get stable around the desired queue size.

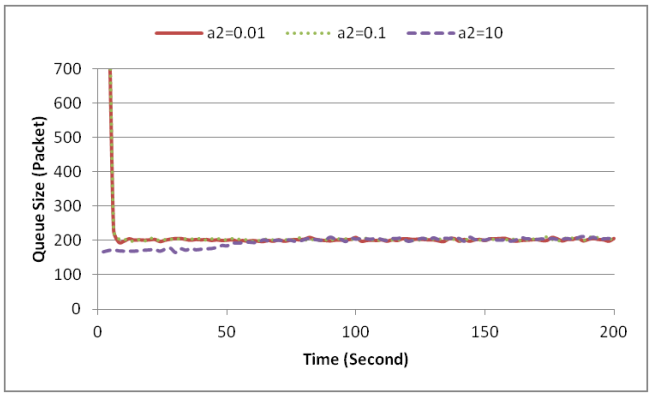


Fig I. 6: More Experiments for the RB-STU Controller Under Wider Range of a_2

Fig I.6 shows more experiments when using wider range of a_2 to be able to give recommendation on W_T weight selection. One can see that the controller is able to have reasonable performance while using small a_2 such as a_2=0.1 and 0.001. However, larger a_2 leads to large rise/fall time. According to Fig. I.2 and Fig. I.3, the controller cannot have small rise/fall time while a_2 is greater than 4.

Appendix J
Linear Matrix Inequality[ScWe04]

A linear matrix inequality is an expression in form of

$$F(x) := F_0 + x_1F_1 + \cdots + x_nF_n \ < 0 \qquad \text{(J.1)}$$

where

- $x = (x_1, \dots, x_n)$ is a vector of n real numbers called decision variables.
- $F_0, \dots., F_n$ are real symmetric matrices, i.e, $F_j = F_j^T \ for \ j = 0, \dots, n$.
- The inequality <0 in eqn. (J.1) means "negative definite". That is $u^T F(x)u < 0$ for all non-zero real vectors u. Because all eigenvalues of a real symmetric matrix are real, eqn. (J.1) is equivalent to saying that all eigenvalues $\lambda(F(x))$ are negative. Equivalently, the maximal eigenvalue $\lambda_{max}\big(F(x)\big) < 0$.

  In other words, A linear matrix inequality (LMI) is an equality

$$F(x) < 0 \qquad \text{(J.2)}$$

where *F* is an affine function mapping a finite dimensional vector space χ to either $\mathbb{H}, \mathbb{S}$ (Hermitian and symmetric space).

In most control applications, LMI's arise as functions of matrix variable rather that scalar valued decision variable. This means that we consider inequalities of the form of eqn. (J.2) where $\mathcal{X} = \mathbb{R}^{m_1 \times m_2}$ is the set of real matrices of dimension $m_1 \times m_2$. A simple example with $m_1 = m_2 = m$ is the Lyapunov inequality $F(x) = A^T X + XA + Q < 0$ where $A, Q \in \mathbb{R}^{m_1 \times m_2}$ are assumed to be given and X ia the unknown matrix variable of dimension $m \times m$. Note that this defines an LMI only if $Q \in \mathbb{S}^m$. We can view this LMI as a special case of eqn. (J.1) by defining an arbitrary basis $e_1, \dots, e_2$ of χ and expanding $X \in \chi$ as $X = \sum_{j=1}^{n} x_j e_j$. Then,

$$F(X) = F\big(\sum_{j=1}^{n} x_j e_j\big) = F_0 + \sum_{j=1}^{n} x_j F(e_j) = F_0 + \sum_{j=1}^{n} x_j F_j \qquad \text{(J.3)}$$

Which is of the form eqn. (J.1). The coefficients x_j in the expansion of X define the decision variables. The number of (independent) decision variables *n* corresponds to the dimension of χ.