uOttawa

L'Université canadienne
Canada's university

FACULTÉ DES ÉTUDES SUPÉRIEURES ET POSTOCTORALES

uOttawa

L'Université canadienne
Canada's university

FACULTY OF GRADUATE AND POSDOCTORAL STUDIES

Qiang Chen
AUTEUR DE LA THÈSE / AUTHOR OF THESIS

Ph.D. (Electrical Engineering)
GRADE / DEGREE

School of Information Technology and Engineering
FACULTÉ, ÉCOLE, DÉPARTEMENT / FACULTY, SCHOOL, DEPARTMENT

Active Queue Management Methods in Computer Communication Networks Based on Pole Placement and H-infinity Optimal Control

TITRE DE LA THÈSE / TITLE OF THESIS

Oliver Yang
DIRECTEUR (DIRECTRICE) DE LA THÈSE / THESIS SUPERVISOR

CO-DIRECTEUR (CO-DIRECTRICE) DE LA THÈSE / THESIS CO-SUPERVISOR

EXAMINATEURS (EXAMINATRICES) DE LA THÈSE / THESIS EXAMINERS

Miodrag Bolic

Jiying Zhao (absent)

Chang Cheng Huang

Ahmed El-Sayed Kamal

Gary W. Slater
Le Doyen de la Faculté des études supérieures et postdoctorales / Dean of the Faculty of Graduate and Postdoctoral Studies

Active Queue Management Methods in Computer Communication Networks Based on Pole Placement and $\mathcal{H}_\infty$ Optimal Control

By

Qiang Chen

A thesis submitted to
the Faculty of Graduate and Postdoctoral Studies
in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy

Ottawa-Carleton Institute for Electrical and Computer Engineering
School of Information Technology and Engineering
Faculty of Engineering
University of Ottawa
Ottawa, Ontario, Canada
August 2006

Your file Votre référence
ISBN: 978-0-494-18577-3
Our file Notre référence
ISBN: 978-0-494-18577-3

Abstract

This thesis studies the control of TCP traffic in the Internet nowadays. Both classical and modern control theory are used to design the controllers for the TCP/AQM system.

For the classical control theoretical approaches, we revisit the simple pole placement technique, and exploit this technique to propose four kinds of controllers for active queue management (AQM) in IP routers: the Proportional controller based on Pole Placement (the P_PP controller), the Proportional-plus-Integral controller based on Pole Placement (the PI_PP controller), the Self-Tuning Proportional controller based on Pole Placement (the ST_P_PP controller), and the Self-Tuning Proportional-plus-Integral controller based on Pole Placement (the ST_PI_PP controller). The transient performance indices - the damping ratio ξ and the undamped natural frequency ω_n are appropriately chosen such that: a) the transient response of the system is satisfied; b) all the poles lie in the left-half of the s-plane to guarantee the stability of the system. The self-tuning controllers (the ST_P_PP and the ST_PI_PP) can maintain on-line tuning of the controller parameters so that good system performance can be ensured even under significant load changes.

This thesis also investigates the application of a robust modern control technique – $\mathcal{H}_\infty$ optimal control to the AQM controller. Two $\mathcal{H}_\infty$ loop-shaping techniques, the S/U Mixed Sensitivity Problem (MSP) and the $S/T/U$ MSP are applied to the controller design. Different weight functions and their impacts on the system performance are discussed and evaluated.

Finally, by modeling the uncertainties in the TCP/AQM system in terms of the uncertainty weight, this thesis also designs a robust AQM controller based on a modern control technique – robust μ analysis. The controller is synthesized by approximating the μ–optimal control problem with the $\mathcal{H}_\infty$ $S/T/U$ MSP. The NP (Nominal Performance), the RS (Robust Stability) and the RP (Robust Performance) of the system under the synthesized controller are analyzed using the μ analysis technique.

Acknowledgements

I would like to thank Prof. Oliver W. W. Yang, my thesis supervisor, for giving me a chance to become a member of his team of researchers, for his persistent support and guidance through the course of this work. Especially, I would like to express my sincere appreciation of his great advice and feedback.

I am also grateful to the committee members of my PhD program, for their insightful comments on the dissertation, and for their precious time and efforts as the committee members.

All the members from the Computer Communication Network Research (CCNR) laboratory have shared years of ideas, arguments, and good times with me.

I gratefully acknowledge the financial support of the Natural Sciences and Engineering Research Council of Canada (Scholarship NSERC PGS B), Ontario Ministry of Education and Training (Scholarship OGS), and the University of Ottawa (Admission Scholarship, Excellence Scholarship, National Excellence Scholarship).

Finally, a special gratitude goes to my parents for their unfailing love and support throughout the years.

Table of Contents

List of Figures

List of Tables

Table of Acronyms and Abbreviations

		Section of 1st Appearance
AQM	Active Queue Management	1.2
AVQ	Adaptive Virtual Queue	1.2.1
BLUE	Blue Algorithm	1.2.1
DRED	Dynamic Random Early Detection	1.2.1
ECN	Explicit Congestion Notification	1.1
ERD	Early Random Drop	1.2.1
FCFS	First Come First Serve	1.1
FIFO	First In First Out	1.1
FRED	Fair Random Early Drop	1.2.1
GPS	Generalized Processor Sharing	1.1
IETF	the Internet Engineering Task Force	1.2.1
MSP	Mixed Sensitivity Problem	1.6
MU_OPT	μ–optimal controller	1.7
NP	Nominal Performance	1.4
NS	Nominal Stability	6.1
P	Proportional controller	1.2.1
PI	Proportional-plus-Integral (PI) controller	1.2.1
PI_PP	Proportional-plus-Integral controller based on Pole Placement	1.7
P_PP	Proportional controller based on Pole Placement	1.7
QoS	Quality of Service	1
RE	plant Relative Error	6.2.3
RED	Random Early Detection	1.2.1
REF_QS	Reference Queue Size	3.1.3
RIO	RED In and Out	1.2.1
RP	Robust Performance	1.4
RS	Robust Stability	1.4
RTT	Round trip time	1.2
SACK	Selective Acknowledgments	1.1
SRED	Stabilized Random Early Detection	1.2.1
SISO	Single-Input Single-Output	App. A
ST_PI_PP	Self-Tuning Proportional-plus-Integral controller based on Pole Placement	1.7
ST_P_PP	Self-Tuning Proportional controller based on Pole Placement	1.7
TCP	Transmission Control Protocol	1
WFQ	Weighted Fair Queuing	1.1
WM	The uncertainty weight W_M	6.2.3
WRED	Weighted Random Early Detection	1.2.1
w.r.t.	with respect to	3.2
WRR	Weighted Round Robin	1.1

Table of Notations and Symbols

		Section of 1st Appearance
a, b	Coefficients of the characteristic equation of the TCP/AQM system	3.1.1
c	Coefficient of the characteristic equation of the TCP/AQM system	3.3.1
dl	Increment parameter in BLUE	1.2.1
$d2$	Decline parameter in BLUE	1.2.1
$f(s)$	Stable scalar transfer function	App. B.1
f_S	Sampling frequency	2.3.3
i	Index denoting the i_{th} TCP source or destination	2.3.1
m	A real number determining the position of the third root of the characteristic equation of the TCP/AQM system	3.3.1
m_1, m_2, m_3	Parameters in W_T	5.4
max_{p}	Maximum drop probability in RED	1.2.1
max_{th}	Maximum queue threshold in RED	1.2.1
min_{th}	Minimum queue threshold in RED	1.2.1
p	Probability of packet drop	2.3.3
p_0	Packet drop probability at the operating point	2.3.3
p_a	Estimation of p_0	2.2
$p_{current}$	Current packet dropping probability	2.2
p_{m}	Congestion notification rate in BLUE	1.2.1
$p_h, p_l, \alpha_h, \alpha_l, \eta_h, \eta_l$	Parameters in the probability distribution function for the think time in the short-lived TCP flow model	2.3.2
$p_{previous}$	Previous packet dropping probability	2.2
$p(x)$	Packet drop function in RED	1.2.1
q	Queue length	2.3.3
q_0	Reference queue size	2.3.3
q_{avg}	Average queue estimate in RED	1.2.1
r_0	Relative uncertainty at steady state in W_M	6.2.1
r_∞	Magnitude of the uncertainty weight at high frequency in W_M	6.2.1
s	Complex variable in the Laplace transformation	2.2
s_1, s_2	Poles of the closed-loop TCP/AQM system	3.1.1
s_3	Pole of the closed-loop TCP/AQM system	3.3.1
t	Time	2.3.2
u	Control signals	5.1
u_Δ	Output of the scaled uncertainty	6.1
v	Controller inputs	5.1
w	Exogenous inputs	5.1
w_{q}	Queue weight in RED	1.2.1
$\dot{x}$	Time-derivative of x	App. D

y_Δ	Input of the scaled uncertainty	6.1
z	Variable in the z-transformation, only in Section 2.3.3	2.3.3
	Exogenous outputs in any Section except 2.3.3	5.1
z_1, z_2	Exogenous outputs	5.1
z_3	Exogenous output	App. B.2.3
A	Maximum steady-state tracking error	5.2
$A, B_1, B_2, C_1, C_2, D_{11}, D_{12}, D_{21}, D_{22}$	State-space realization terms[1] of G as described in (B.5)	App. B.2
B	Transfer function of the low-pass filter in RED	1.2.1
C	Link capacity[2] (packet/second)	2.1
$C, C(s)$	Transfer function of the controller	2.2
$\overline{C}$	Nominal value for C (link capacity)	6.2.1
C_K	Intermediate variable in (B.8)	App. B.3
C_{opt}	$\mathcal{H}_\infty$-norm optimized controller	5.1
D	Disturbance imposing on the control system	App. B.1
D_i	A destination corresponding to S_i	2.3.1
$F(s)$	Closed-loop transfer function of a control system	App. A
$F_l(G, C)$	Lower linear fractional transformation of G and C	App. B.2
$F_u(N, \Delta_M)$	Upper linear fractional transformation of N and Δ_M	App. C.2
G	Generalized plant model	5.1
G_{11}	Transfer function from w to z in the generalized plant model	App. B.2
G_{12}	Transfer function from u to z in the generalized plant model	App. B.2
G_{21}	Transfer function from w to v in the generalized plant model	App. B.2
G_{22}	Transfer function from u to v in the generalized plant model	App. B.2
H	Stack (matrix or vector) of mixed sensitivities	App. B.1
$\mathcal{H}_\infty$	H infinity optimal control	1.4
I	Identity matrix	App. B.2.1
J	$1/J$ is the frequency at which the relative uncertainty reaches 100% in W_M	6.2.1
K	Damping ratio ξ, used in figures	3.2
K_I	Integral gain	1.6
K_P	Proportional gain	1.6
L	Open loop gain	App. B.1
M	Maximum peak magnitude of S	5.2
N	i) Number of long-lived connections (Except in App. C and part of Section 6.1)	2.1
	ii) Lower linear fractional transformation of G and C	6.1
$\overline{N}$	Nominal value for N (number of long-lived connections)	6.2.1

[1] A is also used to denote the maximum steady-state tracking error, and this can be easily determined by the context.

[2] C is also used to denote the controller, and this can be easily determined by the context.

N_{11}	Transfer function from u_Δ to y_Δ	App C.2
N_{12}	Transfer function from w to y_Δ	App C.2
N_{21}	Transfer function from u_Δ to z	App C.2
N_{22}	Transfer function from w to z	App C.2
N_{mix}	Mixed sensitivity matrix	6.1
OM	Undamped natural frequency ω_n, used in figures	3.4
$P\{F > f\}$	Probability distribution function for the file size in the short-lived TCP flow model	2.3.2
$P\{T > t\}$	Probability distribution function for the think time in the short-lived TCP flow model	2.3.2
P, $P(s)$	Plant transfer function in a control system	5.1
$P_1(s)$	Transfer function representing $P_{tcp}(s) \times P_{queue}(s)$	2.2
P_p	Perturbed plant model	6.1
$P_{queue}(s)$	Transfer function representing the queue dynamic in the TCP/AQM system	2.2
$P_{tcp}(s)$	Transfer function representing the TCP dynamic in the TCP/AQM system	2.2
Q	Any stable proper transfer function such that $\|\|Q\|\|_\infty < \gamma$	App. B.3
R	Round trip time	App. D
R_0	Constant round trip time	2.1
$\overline{R}_0$	Nominal value for R_0	6.2.1
R_1, R_2	Routers in the bottleneck configuration	2.3.1
RE/WM	Ratio of the plant relative error divided by uncertainty weight W_M	6.2.3
S	Sensitivity function	5.1
S_i	A TCP-Reno source	2.3.1
T	i) Complementary sensitivity function (Except in 2.3.3)	5.2
	ii) Sample period, i.e. $1/f_S$	2.3.3
T3	Bandwidth with the speed of 44,736,000bps	3.2
T_p	Propagation delay	App. D
U	Input sensitivity function	5.1
W	Window size	App. D
$\overline{W}$	Maximum window size	App. D
W_0	Window size at the operating point	App. D
W_M	Multiplicative uncertainty weight	6.1
W_S	Sensitivity weight function	1.7
W_T	Complementary sensitivity weight function	5.4
W_U	Input sensitivity weight function	1.7
X_∞, Y_∞, F_∞, L_∞, Z_∞	Intermediate variables in the state-space solution of the $\mathcal{H}_\infty$ sub-optimal control problem	App. B.3
Y	Control system output	App. B.1
α, η	Parameters[3] in the probability distribution function	

[3] α is also used to denote the damping factor in AVQ algorithm, and this can be easily determined by the context.

	for the file size in the short-lived TCP flow model	2.3.2
α	Damping factor in AVQ algorithm	3.2
δ	Sampling period in RED	1.2.1
δp	Variation of the probability of packet drop around the operating point	2.2
δq	Variation of the queue length around the operating point	2.2
γ	i) Upper bound[4] of the $\mathcal{H}_\infty$ norm in the $\mathcal{H}_\infty$ sub-optimal control problem	6.1
	ii) Desired utilization in AVQ algorithm	3.2
γ_{min}	Minimized value of γ in case i)	App. B.3
$\lambda_i(\boldsymbol{x})$	The i_{th} eigenvalue of matrix $\boldsymbol{x}$	App. B.3
μ, Mu	Structured singular value	1.3
$\rho(\boldsymbol{x})$	Spectral radius of matrix $\boldsymbol{x}$	App. B.3
$\overline{\sigma}(\boldsymbol{x})$	Maximum singular value of matrix $\boldsymbol{x}$	App. B.1
ω_B	Minimum bandwidth frequency	5.2
ω_n	Undamped natural frequency	1.4
ω_{n0}	Initial value for the undamped natural frequency ω_n	4.3.3
ω_{n1}, ω_{n2}	Boundary values for the preset interval for ω_n	4.3
ξ	Damping ratio	1.4
ξ_0	Initial value for the damping ratio ξ	4.3.3
ξ_1, ξ_2	Boundary values for the preset interval for ξ	4.1
Δ_M	Scaled multiplicative uncertainty	6.1
Δ_S	A fictitious uncertainty block	App. C.3
$\Gamma(\boldsymbol{x})$	Gamma function	2.3.2
Ω	A variable used in the definition of μ	App. C.3
$\boldsymbol{x}^T$	Transpose of matrix $\boldsymbol{x}$	App. B.3
$\|\boldsymbol{x}\|_\infty$	$\mathcal{H}_\infty$ norm	App. B.1

[4] γ is also used to denote the desired utilization in AVQ algorithm, and this can be easily determined by the context.

Chapter 1 Introduction

Originally the Internet was simply designed for packet delivery, and all packets were treated equally without any discrimination or explicit delivery guarantees. This model is called the best effort service model [GeCr01]. In this model, the network exerts its best effort to deliver the packets injected into it without committing to any quantitative quality of service (QoS) bounds, such as the bounds on the throughput, the end-to-end delay, and the packet loss ratio, etc. The users did not request or need any permission to send the packets before transmitting. Therefore the network performance was determined not only by the network itself, but also by the users' offered load, which leads to a complete lack of isolation and protection for network and its performance. The simplicity of this best effort service model certainly played a key role in the deployment of a ubiquitous Internet service. The applications and protocols in this model had been considered to be flexible, adaptive, and robust enough to operate under a wide range of network conditions without requiring any particularly well-defined service. However, a completely uncontrolled network may suffer from congestion collapse, which occurs when the offered load locally exceeds the available bandwidth. Such collapses already occurred in the mid-'80s, when the Internet did not deploy congestion avoidance mechanisms. Repeated congestion problems triggered the definition and implementation of congestion control functions in Transmission Control Protocol (TCP) [Jaco88]. Since then lots of congestion control schemes for the Internet have been proposed.

1.1 Review of Congestion Control Mechanisms

Because of the decentralized characteristic of the Internet, the resource management and control in the Internet naturally involve both end-to-end and local (per-link) decisions. Based on the location where the congestion control mechanisms are implemented, these mechanisms can be divided into two broad classes: *host-based* and *router-based* mechanisms [MaSe97].

The *host-based mechanisms* are usually implemented in the end-hosts, such as the sources or destinations, and they are called *end-to-end flow control.* These mechanisms are based on the Internet-architecture-design philosophy [Clar88] that all flow-related states should be kept on the end hosts. The basic idea is that upon detection of congestion the

sources should inject their packets into the network more slowly, either by reducing their sending window sizes (e.g., in the window-based flow control like TCP), or by decreasing their sending rates (e.g., in the rate-based flow control utilized in the ATM network [OhMu95, BoFe95, ZhYa97, ZhYa00, ZhYa01], or in the TCP-friendly congestion control schemes [WiDe01, HaFl03]). Currently the most widely used end-to-end flow control schemes in the Internet are various versions of TCP protocols such as Tahoe TCP [Jaco88], Reno TCP [Jaco90, Stev94, Stev97], New-Reno TCP [Hoe95, ClHo95, FlHe04], Selective Acknowledgments (SACK) TCP [Floy96a, Floy96b, MaMa96a], Forward Acknowledgment (FACK) TCP [MaMa96b], TCP Vegas [BrMa94], etc. Based on simulations, [FaFl96] compares the performance of different TCP versions of Tahoe, Reno, New-Reno and SACK. The essential operations of the TCP protocol, including the four TCP congestion control algorithms: slow start, congestion avoidance, fast retransmit and fast recovery, can be found in [AlPa99].

In order for a host to detect congestion, the routers must be able to provide the information that the network is currently (or is about to) overloaded; this mechanism is called *feedback*. This mechanism inherently involves both the routers that generate the congestion signals and the receiver hosts that propagate the signal back to the sender hosts for interpreting them accordingly. Therefore the closed-loop flow control mechanisms and overall network performance rely heavily on feedback. There are two forms of feedback: implicit feedback and explicit feedback. In the implicit feedback, the end-hosts do not receive the feedback information from the network (routers) directly; instead they infer the state of the network (congestion or not) from indications by monitoring the performance (e.g. delay, loss, etc) of their own transmissions. The most commonly used implicit signal in the Internet routers is packet drop informed by timeout. Other methods of implicit feedback were proposed, including the observation of the rate at which packets emerge from the bottleneck [Kesh91], and the measurement of the change in end-to-end delay as the transmission rate changes [BrMa94]. In the explicit feedback, the network sends the explicit feedback information to the end-hosts, either in the form of congestion notification or rate indication. One of the explicit feedback methods proposed for TCP/IP network is the Explicit Congestion Notification (ECN) scheme [RaFl99]. In this scheme, the router sets a bit in the packet header (CE bit) whenever it detects incipient congestion. The receiver copies this bit

into the header of the acknowledgment packet, and the flow control mechanism at the sender is responsible for adjusting the sending window accordingly. Another method of explicit feedback is the explicit rate indication [ChCl95], in which a switch performs rate allocation and the calculated rate is recorded in the header of a packet that is explicitly communicated back to the sources via the receiver. This method has been used in ATM networks but not in the Internet.

The end-to-end TCP flow control has been widely used in the current Internet to prevent congestion collapse. However, as the Internet size expands rapidly and the demand for QoS becomes stronger, it is no longer possible to exclusively rely on end hosts to perform end-to-end congestion control. There has been a growing recognition within the Internet community that the network itself must participate in congestion control and resource management. Therefore the introduction of router mechanisms for congestion control that will enable the network to more actively manage its own resources has become inescapable [FlFa99]. Usually the *router-based mechanisms* involve two kinds of schemes: *scheduling* and *queue management*.

Routers use *scheduling* mechanisms to directly manage bandwidth allocation on an output link. Scheduling determines the service order of the packets injected into the routers. The simplest scheduling algorithm is First-Come-First-Serve (FCFS), implemented with a First-In-First-Out (FIFO) queue. An ideal scheduling model is called Generalized Processor Sharing (GPS) [PaGa93], and this ideal model has been emulated by actual scheduling schemes such as Weighted Round Robin (WRR), Weighted Fair Queuing (WFQ) [DeKe89], Worst-case Fair Weighted Fair Queuing (W2FQ) [BeZh96], self-clocked fair queuing [Gole94], and deficit round robin [ShVa95], etc. It has been proved [PaGa94] that there is a bound on the worst case end-to-end delay experienced by a flow that passes through a series of GPS schedulers.

The role of *queue management* is to control the length of the queue and potentially which flows occupy it, by selecting which packets to drop and determining when this is appropriate. We know that packet-drop has been interpreted by the TCP as a congestion indication; thus, the queue management mechanism is also the place where the feedback to the source originates by dropping/marking packets. It has been shown [SuLa98] that, in a congestion situation where the offered load persistently exceeds link bandwidth, and the

sources do not react uniformly to congestion, the queue management discipline has a stronger impact on bandwidth sharing than the scheduling discipline itself. Since the main topic in our thesis is the active queue management, therefore we use a separate section (Section 1.2) to review the queue management schemes in detail.

1.2 Review of Queue Managements

Traditionally the queue management is for congestion recovery, which means the routers start to drop packets only when the congestion has actually occurred, namely when the queue is full. For example, in the classical Drop-Tail method, new packet arrivals are dropped only when the queue overflows. Drop-Tail mechanism has served the Internet well for years, but has two serious disadvantages. The first one is that it sustains full queues, which means it leads to large queuing delays and high loss rates at congested links. The second one is that it can cause lockout due to traffic phase effects. The first one is obvious, and here we explain the second one. In a Drop-Tail router, when a queue overflows, packets from several sources are often dropped and all these sources reduce their transmission rate simultaneously and their control actions become synchronized, and we call this global synchronization. It has been shown [FlJa91] that the global synchronization can lead to reduced link utilization or lockout phenomena where a few sources monopolize the queue space, preventing others from getting in.

Another deterministic queue management mechanism is Drop-Head, in which the gateway drops the packet from the front of the queue when a new packet arrival finds the queue full. It has been shown [LaNe96] that the Drop-Head scheme can improve the performance of TCP by allowing the congestion indication signal to reach the sender faster rather than waiting for the full queue to be transmitted first as in the Drop-Tail Scheme. It can also prevent the lockout and improve fairness because dropping a packet from the front of the queue leads to a loss distribution that follows the buffer occupancy among connections at the instances when the queue is full.

As an improvement over the Drop-Tail or Drop-Head method, Random Drop gateway randomly drops a packet from the gateway queue when a new packet arrives at a full gateway buffer. Since a packet randomly and uniformly selected from all incoming traffic will belong to a particular flow with a probability proportional to the bandwidth share of that flow on that gateway, therefore the gateway can send congestion signals to those users whose

traffic contributes more to the congestion of the router. And thus it can achieve improved fairness for late-starting connections and slightly improved throughput for connections with longer round trip times (RTTs) [Hash89]. One of the disadvantages of the Random Drop scheme is that the operation of randomly selecting a packet to drop can be computationally expensive.

The Drop-Head scheme and the Random Drop algorithm solve the problem of lockout but do nothing to solve the problem of the full queue. That is because they do not react to congestion fast enough but only after congestion has occurred. The only way to solve the full queue problem is to drop packets proactively rather than reactively. This is usually called Active Queue Management (AQM), in which the routers are allowed to control which packets to drop and when this should happen in order to avoid congestion. Therefore the AQM is for congestion avoidance rather than congestion recovery.

1.2.1 Active Queue Management

The Internet Engineering Task Force (IETF) has proposed the deployment of AQM mechanisms [Brad98] at gateways to improve the performance of TCP congestion control. By "active queue management ", we mean core routers inside networks are equipped with the capability to detect incipient congestion and to explicitly/implicitly signal traffic sources by marking/dropping packets before congestion actually occurs. AQM has multiple goals: a) to improve the congestion avoidance strategy of the TCP protocol; b) to ensure high network utilization by reducing the packet loss due to buffer overflow; c) to reduce the average queue length in gateways and thus to decrease the end-to-end delay experienced by packets; d) to avoid the problem of global synchronization; e) to eliminate the bias against bursty traffic; f) to penalize aggressive flows.

AQM has been a very active research area in the Internet community. Early Random Drop (ERD) [Hash89] is the first generation of AQM algorithm. It is a congestion avoidance scheme that initiates the drop of packets when congestion is anticipated instead of only when the queue becomes full. The congestion detection is implemented in the simplest form by a static threshold in queue length, although more elaborate measures like exponentially weighted moving averages or link utilization have been proposed. The control interval is chosen in terms of packet arrivals, and the drop probability is fixed. Therefore ERD drops each packet arrival at the gateway with a fixed drop probability if the queue length exceeds a

certain drop level. ERD gateways are certainly an improvement over drop tail because they alleviate to a great extent the lockout problem. However ERD has been shown [Zhan89] to be unsuccessful in controlling misbehaving users, and to be biased against bursty traffic. This bias occurs because the contents of the queue do not necessarily reflect the average traffic.

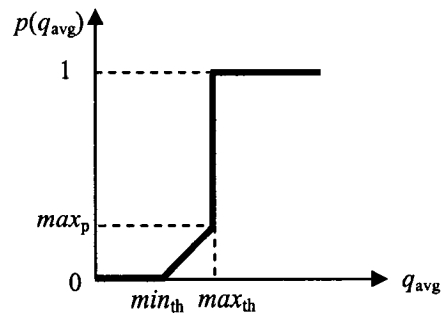

Fig. 1.1: RED drop function

In order to overcome the shortcomings of ERD, Ref. [FlJa93] proposes a new AQM algorithm called Random Early Detection (RED). The RED gateways attempt to mark packets sufficiently frequently to control the average queue size and avoid the biases described above. This scheme works as follows. The RED maintains an exponentially weighted moving average of the queue length which it uses to detect congestion. When the average queue length exceeds a minimum threshold (min_{th}), arriving packets are dropped or marked with a calculated probability. This probability depends on the average queue length, the time elapsed since the last packet was dropped, and a preset probability parameter – the maximum drop probability (max_p). When the average queue length exceeds a maximum threshold (max_{th}), all arriving packets are dropped or marked. There are two distinct algorithms operating in a RED gateway:

1) The average queue size computation, which uses an exponentially weighted moving average. It has been shown [MiGo00] that the average queue length is actually the output of a first-order low-pass filter with a transfer function of $B/(B+s)$, where $B = -\frac{\log_e(1-w_q)}{\delta}$, with w_q being the queue weight, and δ being the sampling period. This low-pass filter reflects the design goal of RED: to track the average queue size (low frequency signal), and to filter out bursts (high frequency signal). The input to this filter is the instantaneous queue length, and the output is the average queue estimate q_{avg}.
2) The algorithm for calculating the packet drop probability. This algorithm can be

described by a packet drop function $p(x)$ that takes as its argument the above calculated average queue length q_{avg} at the router. Fig. 1.1 illustrates the form of $p(x)$. Namely we have:

$$p(q_{avg}) = \begin{cases} 0, & 0 \le q_{avg} \le min_{th} \\ \dfrac{q_{avg} - min_{th}}{max_{th} - min_{th}} max_p, & min_{th} \le q_{avg} \le max_{th} \\ 1, & max_{th} < q_{avg} \end{cases}$$

The RED is the most prominent and widely studied AQM mechanism and successfully addresses the problems found in its predecessors. It can prevent global synchronization, reduce packet loss ratios and minimize the bias against bursty sources. However, studies such as [BoMa00] [ChJe00] [FiBo00] have shown that it is very difficult to tune the RED parameters in order to perform well under different traffic conditions. The parameter settings are based on heuristics, and the proposed configuration is suitable only for the particular traffic conditions studied.

BLUE [FeSh02] uses packet loss and link under-utilization events to adjust the rate of congestion notification. The congestion notification rate p_m (packet drop probability) is increased at a preset rate if the queue size exceeds a threshold, and it is decreased if the link is idle. The amount of increase is *d1*, and decrease *d2*, at a rate of *1/freezetime*. BLUE behaves well if operated within a region of round trip time (RTT) and number of connections N for which the parameters *d1* and *d2* were set. However, changes in the dominant RTT of the connections going through the queue, or a significant change in N can invalidate the parameter settings and lead to queue backlog oscillation between loss and under-utilization. The amount of change of notification rate p_m during a queue full or link idle event, is the product of the time spent in this event multiplied by the rate of change *d(1,2)/freezetime*. This time is related to the delay in the response of the TCP sources to the changed notification rate. The greater the RTT, the greater will be the p_m adjustment. If the RTT increases, so does the change in p_m and this may result in backlog oscillation. Another cause of instability is a large change in the number of connections. This instability is the result of the adjustment of congestion notification rate p_m by a constant *d1* or *d2*, despite the nonlinear relation of p_m and N. Recall that based on the TCP Friendly Equation [FlFa97] the function of p_m versus N requires larger changes of p_m for larger N.

Adaptive Virtual Queue (AVQ) [KuSr04] maintains a virtual queue whose capacity (called virtual link capacity) is less than the actual capacity of the link. When a packet arrives in the real queue, the virtual queue is also updated to reflect the new arrival. Packets in the real queue are marked/dropped when the virtual buffer overflows. The virtual link capacity at each link is then adapted to ensure that the total flow entering each link achieves a desired utilization of the link. AVQ is a primarily rate-based control scheme, as opposed to queue length-based control scheme. This provides early feedback. The major deficiency of this scheme is that it does not have the capability to clamp the steady-state queue size to a pre-set reference value.

Some other AQM algorithms such as *Fair Random Early Drop* (FRED) [LiMo97], *Dynamic Random Early Detection* (DRED) [AwOu01], *Stabilized Random Early Detection* (SRED) [OtLa99], *Self-Configuring Gateway* [FeKa99], *Yellow* [LoZh05], *Weighted Random Early Detection* (WRED) [Cisc98] and *RED In and Out* (RIO) [ClFa98] have been proposed. Most of these are heuristic algorithms and very few systematic and comparison (e.g. [ZhYa02]) were done until recently.

Recently, there has been systematic work on the theoretical analysis of AQM algorithms. In [MiGo00], Vishal Misra et al developed a nonlinear dynamic model of TCP/AQM using fluid-flow and stochastic differential equation analysis. In [HoMi01a], they converted the nonlinear model to a linear system via the technique of linearization, and further proposed the block diagram of the linearized TCP/AQM control system. Based on the classical control theory and this linearized TCP/AQM model, a proportional (P) controller and a proportional-plus-integral (PI) controller for AQM were designed [HoMi01b], [HoMi02]. Although system stability can be achieved under unchanged traffic conditions, the parameter values of these controllers are fixed according to the initial settings of the network and traffic conditions, and are not tunable when the network and traffic conditions vary. Therefore, under certain configurations these P controller or PI controller may exhibit sluggish response, or even become unstable if significant load changes occur. A self-tuning AQM controller was presented in [ZhHo03], but the controller always self-tunes even when traffic load changes slightly, which may often be unnecessary.

To shape the closed-loop dynamics of the TCP/AQM system, researchers start to design the AQM controller based on modern robust control theory. For example, the $\mathcal{H}_\infty$

controller for the TCP/AQM system [QuOz04] uses a frequency domain solution [ToOz95] to synthesize the controller. But their solution does not consider the shaping of the input sensitivity function, and their controller is complicated, making it difficult to implement in the real network. Although performance evaluation in [QuOz04] was done in *simulink* simulations, the theoretical model equation, instead of the real TCP protocol, is simulated, making any real-life applications much less convincing.

1.3 Review of the Control Theory

Automatic control has played a vital role in the advance of engineering and science. There have been many developments in automatic control theory during recent years. It is difficult to provide an impartial analysis of an area while it is still developing; however, looking back on the progress of feedback control theory it is by now possible to distinguish some main trends and point out some key advances.

In the *primitive period* of automatic control (before early 1900's), the first significant work in automatic control was James Watt's centrifugal governor for the speed control of a steam engine in the eighteenth century. In 1868, J.C. Maxwell provided the first rigorous mathematical analysis of a feedback control system, which was a milestone that signified the control theory as an independent discipline.

The period from early 1900's until 1960 is called the *classical period*, in which some important classical methods are developed. From 1900 to 1940, some significant works were due to Minorsky (1922, determining stability from the differential works), Nyquist (1932, developing a simple procedure for determining the stability of closed-loop systems), and Hazen (1934, introducing the design of relay servomechanisms capable of closely following a changing input). During the decade of the 1940s, *frequency-response methods* [Bode40], [Macc45] made it possible for engineers to design linear closed-loop control systems that satisfied performance requirements. In [Bode40], H.W. Bode used the magnitude and phase frequency response plots of a complex function, and investigated closed-loop stability using the notions of *gain and phase margin*. From the end of 1940s to early 1950s, the *root-locus method* due to Evans was fully developed [Evan48]. This method provides a direct way to determine the closed-loop pole locations (*pole placement*) in the s-plane. Subsequently, during the 1950's, much controls work was focused on the s-plane, and on obtaining desirable closed-loop step-response characteristics in terms of rise time, percent overshoot,

and so on. The *frequency-response* and *root-locus* methods, which are the core of classical control theory, lead to systems that are stable and satisfy a set of more or less arbitrary performance requirements. Such systems are, in general, acceptable but not optimal in any meaningful sense. Sine the late 1950s, the emphasis in control design problems has been shifted from the design of one of many systems that work to the design of one optimal system in some meaningful sense.

The period from 1960 through present times is called the *modern period.* During the period from 1960 to 1980, optimal controls of both deterministic [Jaco73] and stochastic systems [Tse71], as well as adaptive and learning control of complex systems [GoRa80, Fu70], were fully investigated. The vital work of Lyapunov in the time-domain control of nonlinear systems [KaBe60] was publicized. The optimal control of systems, providing the design equations for the *linear quadratic regulator (LQR)*, was discussed [Kalm60]. The theory of Kalman provided optimal solutions that yielded control systems with guaranteed performance.

With all their power and advantages, the above modern control schemes were lacking in some aspects. The guaranteed performance obtained by solving matrix design equations means that it is often possible to design a control system that works in theory without gaining any engineering intuition about the problem. On the other hand, the frequency-domain techniques of classical control theory impart a great deal of intuition.

Another problem is that a modern control system with any compensator dynamics can fail to be robust to disturbances, unmodelled dynamics, and measurement noise. On the other hand, robustness has been built in with a frequency-domain approach using notions like the pole placement, or the gain and phase margin. Therefore, in the late 1970s and early 1980s, there was a great deal of activity to extend classical frequency-domain techniques and the root locus to multivariable systems, namely to unite the classical control with the modern control. This has led to the era of *robust modern control theory* from late 1970s to the present. Its development is centered on $\mathcal{H}_\infty$ *control* [Zame79], *robust control* (e.g. *robust-μ analysis*) [Doyl82], and their associated topics. In 1981 seminal papers [DoSt81], [SaLa81] showed the importance of the *singular value* plots versus frequency in robust multivariable design. Using these plots, many of the classical frequency-domain techniques can be

incorporated into modern design. Their work shows that these *robust modern control* schemes blend the best features of classical and modern techniques.

1.4 Motivations

The importance of AQM has been recognized in the Internet research community for a long time. As mentioned in Section 1.2.1, the RED and lots of its heuristic variant algorithms are not good enough to handle all kinds of network scenarios. Therefore in this work, instead of using heuristic algorithms, we focus on applying the control theoretical approaches to design the AQM controllers, with the intention to achieve better TCP/AQM performance.

Unlike the classical P controller and the PI controller [HoMi02] that have been designed to achieve system stability, we would like to have controller parameters that are not determined solely by the network and traffic conditions, but also to allow the user much freedom in determining the controller parameters. This motivates us to use the pole placement technique in the classical control theory. Because when the pole placement approach is used, the user can specify the damping ratio ξ and/or the undamped natural frequency ω_n, which are the typical transient performance indices, to design the controller. Appropriately chosen ξ and/or ω_n can ensure the stability and transient performance of the TCP/AQM system.

Under the above pole placement controllers or any other controllers with fixed parameters (e.g., the P controller and the PI controller in [HoMi02]), when significant load changes occur in the network, the system will show sluggish response or even become unstable if the controller parameters are not on-line tunable. Therefore it would be desirable to incorporate the self-tuning feature into the controller design based on the pole placement technique.

It is interesting to note that, although so far lots of work has been done using the classical control theory, very few are conducted based on the modern robust control theory except the work [QuOz04], in which an $\mathcal{H}_\infty$ controller is proposed for AQM. The advantages of the modern robust control theory such as the optimal $\mathcal{H}_\infty$ control are obvious: the closed-loop dynamics of the system can be shaped explicitly by appropriately chosen weight functions; they can improve the robustness against uncertainties, etc. However the controller

studied in the literature e.g. [QuOz04] is very complicated due to the usage of complex weight functions and a frequency domain solution.

The robust μ analysis is another technique in the modern robust control theory. The NP (Nominal Performance), the RS (Robust Stability) and the RP (Robust Performance) of the system can be quantitatively analyzed when the μ analysis approach is used. At the time of writing this thesis, no work on AQM design has been done based on the robust μ analysis. Therefore as part of the research work, we would like to apply the robust μ analysis to the AQM controller design.

1.5 Objectives

Generally we are interested in applying control theoretical approaches to the AQM controller design. More specifically, the following objectives are focused on in this work

1) To study the use of pole placement as a classical control technique in the AQM controller design.
2) To make the algorithm in 1) on-line tunable such that even under significant load changes, a desirable performance of the TCP/AQM system can still be achieved.
3) To study the use of the modern control approaches such as the $\mathcal{H}_\infty$ loop-shaping technique and the robust μ analysis technique in the AQM controller design.
4) To carry out performance evaluation and comparison for the designed controllers to demonstrate their capability.

1.6 Approaches and Methodologies

To design our AQM controllers using classical control theoretical approaches, we shall first design the proportional controller and the proportional-plus-integral controller based on the pole placement technique of the classical control theory. We choose proportional controller because it is expected to give fast response and it is very simple. For the proportional-plus-integral controller, although it shows slower response than the proportional controller, it can eliminate the steady-state error between the system output and the reference input. We choose the pole placement method because, as mentioned in Section 1.4, we expect it to give the user more freedom on controller parameters, and to ensure good transient performance. This approach can be summarized as follows. For a second order system, the locations of the two poles of the closed-loop function determine the system stability and transient performance. By appropriately choosing the damping ratio ξ and/or the undamped natural frequency ω_n, we want to place the poles at satisfactory places on the s-plane. For a higher

order system, we shall adopt the dominant pole placement design approach, namely we design a controller such that the pair of dominant poles can give rise to a desired damping ratio ξ and an undamped natural frequency ω_n. In this thesis, through mathematical manipulation, we derive the design formula for determining the controller parameters K_P and K_I from the damping ratio ξ and the undamped natural frequency ω_n. More detailed background information on pole placement can be found in Appendix A.

Then we shall design the AQM controllers based on one of the modern robust control schemes - $\mathcal{H}_\infty$ optimal control. Specifically, we adopt the $\mathcal{H}_\infty$ loop shaping techniques, such as the *S/U* Mixed Sensitivity Problem (MSP) and the *S/T/U* MSP. Through the usage of the loop shaping technique, we want to explicitly bound the closed-loop dynamics of the system by the weight functions so that satisfactory AQM performance can be obtained. To solve the MSP, we shall formulate it in the general control configuration and then use the state-space solution [DoGl89]. The details of the MSP and its solution are given in Appendix B. In this thesis, through theoretical analysis and simulations, we will determine the guidelines for choosing the weight functions and thus propose the *S/U* MSP controller and the *S/T/U* MSP controller for AQM routers.

With the MSP controllers, we cannot consider explicitly the uncertainties of the TCP/AQM system, and the robust stability (RS) and robust performance (RP) of the system cannot be analytically evaluated. Therefore we shall design an AQM controller based on another robust modern control approach – robust μ analysis. Using this method, the system uncertainties can be incorporated in the controller design, and we can therefore synthesize a controller that satisfies the robust stability requirement. The details of the robust μ analysis technique will be presented in Appendix C. We use the μ toolbox of the MATLAB to solve the $\mathcal{H}_\infty$ optimal controller. The MATLAB software is also used to draw Bode diagrams, to determine the uncertainty weight in the μ–optimal controller design, and to analyze the NP, RS and RP of the TCP/AQM system quantitatively.

In terms of simulators for computer networks, the two most widely used tools are the Network Simulator (ns-2) and the OPNET modeler. Although ns-2 provides substantial support for simulation of TCP over wired and wireless (local and satellite) networks, it is not a polished and finished product, instead it is the result of an on-going effort of research and development. In particular, bugs in the software are still being discovered and corrected. In

contrast, OPNET Modeler is the industry's leading environment for network modeling and simulation, allowing us to design and study communication networks, devices, protocols, and applications with unmatched flexibility and scalability. Modeler is used by the world's most prestigious technology organizations to accelerate the R&D process. Therefore, we shall implement all designed controllers in OPNET simulations [OPNE01]. The performance of the system with the designed controllers are investigated and evaluated through extensive OPNET simulations with different scenarios, including the case of different time-varying round trip times (RTTs) for different connections. The TCP protocol is implemented, and different TCP sources such as the greedy ftp sources and the burst http sources are used in the simulations. For comparison, the RED algorithm [FlJa93], the P controller and the PI controller [HoMi02] are also implemented in the simulations.

1.7 Contributions

The main contributions of this thesis are:

1) The formulation of the P_PP controller (a Proportional controller based on the Pole Placement), the PI_PP controller (a Proportional-plus-Integral controller based on the Pole Placement), and their performance evaluations. To the best of our knowledge, this is the first time that the pole placement technique is used to the AQM design.
2) The formulation of the ST_P_PP controller (a Self-Tuning Proportional controller based on the Pole Placement), and the ST_PI_PP controller (a Self-Tuning Proportional-plus-Integral controller based on the Pole Placement). We demonstrate how they can ensure that the system can still perform well even under significant load changes.
3) The formulation of the $\mathcal{H}_\infty$ *S/U* MSP controller for AQM and its performance evaluations. We apply the $\mathcal{H}_\infty$ loop-shaping technique to design the AQM controller. Through theoretical analysis and simulations, we are able to provide the guidelines regarding the selection of a different sensitivity function weight W_S, and for the first time, the input sensitivity function weight W_U.
4) The formulation of the MU_OPT (μ-optimal) controller for AQM and its performance evaluation. By the time of writing this thesis, no work of applying the robust μ-analysis technique to the AQM design has been reported in the literature.

1.8 Thesis Organization

The thesis is organized as follows.

Chapter 2 describes the network model and the simulation model used in this thesis. The linearized TCP/AQM network model and its control block diagram are given in this chapter. Chapter 2 also presents the network configuration for OPNET simulations, the TCP traffic flow model, the assumptions used in the simulations, and the implementation of the continuous time controller.

In Chapter 3, we propose and evaluate the P_PP controller and the PI_PP controller. We derive the formulae that relate the controller parameters K_P, K_I with the damping ratio ξ, the undamped natural frequency ω_n and the network parameters. The design procedures for these two controllers are formulated.

The ST_P_PP controller and the ST_PI_PP controller are proposed and evaluated in Chapter 4. Based on the controllers proposed in Chapter 3 and by incorporating the idea of self-tuning, we derive the formulae for on-line monitor of ξ and ω_n, determine the acceptable intervals for ξ and ω_n, and propose the design procedures.

Chapter 5 presents the AQM controller design based on $\mathcal{H}_\infty$ optimal control. The $\mathcal{H}_\infty$ loop-shaping techniques are applied to the TCP/AQM model to design the controllers. Specifically, two methods – the *S/U* MSP and the *S/T/U* MSP are applied to the controller design.

In Chapter 6, we design the AQM controller based on robust μ analysis. Using a graphic technique, we model the uncertainties of the TCP/AQM system in the form of uncertainty weight and then synthesize the controller by approximating the μ–optimal control problem with the $\mathcal{H}_\infty$ *S/T/U* MSP.

Some observations on the usage of the controller algorithms proposed in this thesis and a comparison of those controller algorithms are given in Chapter 7. Chapter 8 concludes the thesis and presents some suggestions of potential areas for future work.

1.9 Publication List

The following is a list of the publications related to this research.

Journal Publications:

1) Qiang Chen, Oliver 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.
2) Qiang Chen, Oliver W. W. Yang, "AQM Controller Design for IP Routers Supporting TCP Flows Based on Pole Placement, " *IEE Proceedings - Communications*, Vol. 151, No. 4, Aug. 2004, pp. 347 – 354.
3) Qiang Chen, Oliver W. W. Yang, "Design of AQM Controllers for IP Routers Based on $\mathcal{H}_\infty$ Optimal Control, " *International Journal of Internet Protocol Technology*, Vol. 1, No. 4, Jul. 2006, pp. 252 – 263.

Conference Publications:

4) Qiang Chen, Oliver W. W. Yang, "On Designing Traffic Controller for AQM Routers Based on Robust μ-Analysis," *Proceeding of the 2005 IEEE Global Telecommunications Conference (GLOBECOM 2005)*, Nov. 18 - Dec. 2, 2005, St. Louis, MO, USA
5) Qiang Chen, Oliver W. W. Yang, "Design of AQM Controller for IP Routers Based on $\mathcal{H}_\infty$ *S/U* MSP," *Proceeding of the 2005 IEEE International Conference on Communications (ICC 2005)*, May 16-20, 2005, Seoul, Korea.
6) Qiang Chen, Oliver W. W. Yang, "On Designing an $\mathcal{H}_\infty$ *S/T/U* MSP Controller for AQM Routers," *Proceeding of the 39th Annual Conference on Information Sciences and Systems 2005 (CISS 2005)*, March 16-28, 2005, Baltimore, USA.
7) Qiang Chen, "A PI Controller for IP Routers Supporting TCP Flows with Pole Placement Design, " *Proceeding of the Twenty-Second Biennial Symposium on Communications*, June 1-3, 2004, Kingston, Ontario, Canada, pp.286-288
8) Qiang Chen, Oliver W. W. Yang, "A ST-PI-PP Controller for AQM Router, " *The 2004 Proceeding of the IEEE International Conference on Communications (ICC 2004)*, June 20-24, 2004, Paris, France, pp. 2277 - 2281
9) Qiang Chen, Oliver 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
10) Qiang Chen, Oliver 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
11) Yang Hong, Qiang Chen, "Design of TCP Traffic Controller for AQM Routers Based on Gain Margin Specification," *Proceeding of the 2003 International Symposium on Performance Evaluation of Computer and Telecommunication Systems (SPECTS 2003),* July 20 - 24, 2003, Montreal, Canada, pp.452-457
12) Yang Hong, Qiang Chen and Oliver W. W. Yang, "A PI Controller for AQM Routers Supporting TCP Flows," *Proceeding of the IASTED International Conference on Applied Modeling and Simulation (AMS 2002)*, November 4-6, 2002, MIT, Cambridge, USA, pp.157-162
13) Yang Hong, Qiang Chen and Oliver 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, Session #: 1537, Session Name: TCP

Chapter 2 Network Model, Operation and Assumptions

In this chapter, we briefly review the TCP/AQM network model that is used to design the AQM controllers in this thesis. The network model considered in this thesis is first given, followed by the linearized control model for the TCP/AQM system. We then present the specific model, operation and assumptions to be used for the rest of the thesis, such as the network configuration for simulations, the TCP traffic flow model, and the implementation of continuous-time controllers.

2.1 Network Model

We consider an end-to-end packet-switching network using TCP/IP to support various applications.

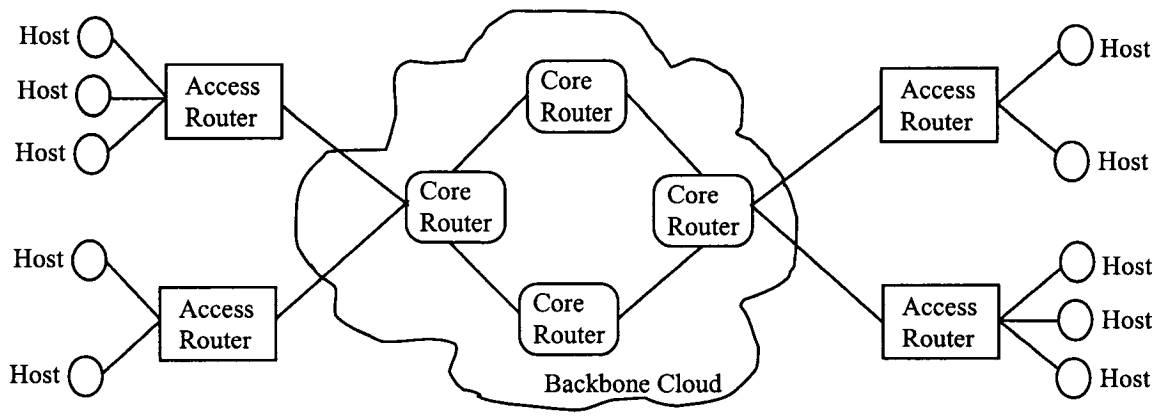

Fig. 2.1: Generic Network Model

Fig. 2.1 illustrates a generic topology for an IP network in our investigation: hosts are attached to access routers, and access routers are interconnected to a backbone cloud formed by core routers. The function of the access/core routers is to carry and forward IP packets from host to host to enable end-to-end communications. This commonly used network model can be found in any textbook on networking, e.g. [Tane88].

The hosts in the network model communicate with each other using TCP/IP protocol with more details provided in Section 2.1.1. The traffic (e.g. by the http application or the ftp application) generated at a source host will go to the destination host through a fixed path consisting of the access router and the core router. Upon receiving the traffic correctly, the

destination host sends the ACK back to the source host. All together, a Round Trip Time (RTT) R_0 is incurred from the moment the source host sends a packet to the destination host until it receives the response (ACK). There are N long-lived TCP connections between the source hosts and the destination hosts. Congestion can occur when the queue builds up at the bottleneck link, and the link capacity at the congested bottleneck link is C (packets/second). Each router (core or access) implements an AQM controller to be studied in this thesis. The AQM controller takes the variations of the queue length as the input, and generates the drop probability as the output, which determines the rate of the packet drops. Since the packet drops serve as the implicit congestion notification signal to the source, the source will adjust its congestion window (sending rate) accordingly, and this in turn will change the queue length.

2.1.1 TCP Operation

The following is a summary of the TCP operation used in our simulation model. Details can be found in [AlPa99]. TCP is a connection-oriented end-to-end acknowledged transport protocol and provides a reliable byte-stream transfer service between two end nodes on the Internet. Since the connection is built between the sender and receiver, the sender sends packets and later it receives the acknowledgement (ACK) from the receiver. Any damage and corruption to the transmitted packets are considered as a failure, and those packets are retransmitted. Because the network keeps the traces of all the packets based on their sequence numbers, it is easy to reorder the packets which may be misordered during the transmission and discard those duplicate packets. TCP can divide a large packet into pieces at the sender based on the window size and integrating those pieces together at the receiver. In summary, the TCP protocol can be divided into four congestion control algorithms: *Slow Start*, *Congestion Avoidance*, *Fast Retransmit* and *Fast Recovery*.

The *Slow Start* occurs whenever the new connection is established or the sender starts to transmit packets after being idle or timing out. The congestion window size *cwnd* is initialized to one packet. Once the sender receives the acknowledgement ACK of the new data packet from the receiver, the sender doubles its congestion window size *cwnd* until *cwnd* reaches the *Slow Start* threshold *ssthresh*. So in the *Slow Start* algorithm the congestion window size *cwnd* increases exponentially. If the congestion occurs and the packet is

dropped before *cwnd* reaches the *ssthresh*, instead of continuously implementing the *Slow Start* algorithm, the *Congestion Avoidance* algorithm will take over.

The *Congestion Avoidance* algorithm is implemented right after *cwnd* is greater than *ssthresh*, or when the congestion occurs before *cwnd* reaches *ssthresh*. The *cwnd* is increased linearly in the *Congestion Avoidance* algorithm – the sender increases its *cwnd* by $1/cwnd$ packet upon receiving the ACK of a new data packet from the receiver. Therefore the increment in *cwnd* is roughly one packet in each round-trip time *RTT*. As the *cwnd* continuously grows above the bandwidth delay product of the network, the packets will be queue up at the buffer in the router. The TCP sender notes that its *cwnd* is too large only after the packet was lost. Once the TCP sender runs into time out, the *ssthresh* is set to half of *cwnd* and enter *Slow Start*.

The traditional TCP sender awaits the retransmission timer to expire in order to retransmit the lost packets. With *Fast Retransmit*, the sender will retransmit the lost packet after receiving several duplicate acknowledgements for the same TCP packet and set the *ssthresh* to the half of *cwnd* and enter *Slow Start*. However, sometime because of different network paths, it may occur that a packet is in flight in the network while its following packets have already reached the destination. Then the duplicate ACKs will be sent to the sender and lead to the TCP sender going back to *Slow Start*. The network may appear under-utilization for a period of time. The new algorithm, called *Fast Recovery*, was then introduced.

With *Fast Recovery*, upon receiving several duplicate ACKs, instead of going to slow-start, the sender retransmits one packet and reduces its *ssthresh* to half of the flight data, which are sent out but not acknowledged data, and congestion window size is set to *ssthresh* plus 3 packets. Fast recovery ends with receiving the next successful ACK that acknowledges new data and set *cwnd* to *ssthresh*.

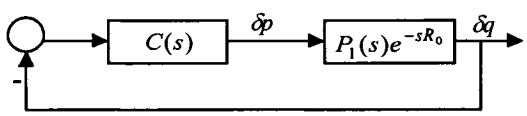

Fig. 2.2: Block–diagram of the linearized TCP/AQM control system

2.2 Control Model of the TCP/AQM System

All the controllers designed in this thesis are based on the TCP/AQM model [HoMi01a] as shown in Fig. 2.2. This model is a linearized one that is obtained through the small-signal linearization of a nonlinear dynamic model [MiGo00]. For completeness, the nonlinear model is given in Appendix D.

In Fig. 2.2, $C(s)$ is the compensator or controller, and $P_1(s)e^{-sR_0}$ is the "plant" or the TCP/AQM system we are trying to control. The parameter R_0 is the round trip time, which causes a delay in the feedback of losses. The parameter δp is the variation of the probability of packet drop around the operating point, and δq is the variation of the queue length around the operating point. Let C be the link capacity (packet/second) and N the load factor (number of long-lived TCP sessions). Let $P_{tcp}(s)$ be the transfer function representing the TCP dynamic in the TCP/AQM system, and $P_{queue}(s)$ be the queue dynamic in the TCP/AQM system. Then,

$$P_{tcp}(s) = \frac{\frac{R_0 C^2}{2N^2}}{s + \frac{2N}{R_0^2 C}}, \qquad P_{queue}(s) = \frac{\frac{N}{R_0}}{s + \frac{1}{R_0}}, \tag{2.1}$$

and the transfer function $P_1(s)$ is given by $P_{tcp}(s) \times P_{queue}(s)$.

The number of active long-lived TCP flows N can be estimated using a simple form of flow cache (called *zombie list*). The zombie list is equivalent to a list of N recently seen flows augmented with a *count* and *timestamp* field [OtLa99]. The link capacity C can be estimated by the service rate of the local congested router [ZhHo03]. The round trip time R_0 can be estimated by the TCP throughput equation $R_0 = \frac{\sqrt{2}N}{C\sqrt{p_0}}$, where p_0 is the equilibrium packet dropping probability and the output of the AQM controller [ZhHo03]. It is not easy to obtain equilibrium packet dropping probability p_0 upon the dramatic change of the network environment, so we use the average value of the previous packet drop probability and the current drop probability (i.e. $p_a=(p_{previous} + p_{current})/2$) to approximate p_0. Therefore the round-trip time can be estimated by $R_0 = \frac{\sqrt{2}N}{C\sqrt{p_a}}$ in the real-time implementation.

By choosing different forms of $C(s)$ and different methods to determine the parameters

of $C(s)$, one obtains the 4 different AQM algorithms (controllers) used in this thesis:

(i) a P controller if $C(s) = K_P$;

(ii) a PI controller if $C(s) = K_P + K_I / s$;

(iii) a $\mathcal{H}_\infty$ optimal controllers if $C(s)$ is determined by applying the $\mathcal{H}_\infty$ loop shaping technique to the AQM controller design; and

(iv) a μ-optimal controller if $C(s)$ is designed based on robust μ analysis.

In this thesis, the proposed the P_PP controller, the PI_PP controller, the ST_P_PP controller and the ST_PI_PP controller determine the controller parameters K_P and K_I based on the pole placement technique of the classical feedback control theory. The proposed *S/U* MSP controller determines $C(s)$ based on the $\mathcal{H}_\infty$ *S/U* MSP loop shaping technique. The proposed MU_OPT controller synthesizes $C(s)$ based on μ optimal control theory.

2.3 Simulation Model

In this thesis, all the designed controllers are evaluated through extensive OPNET simulations [OPNE01]. Here we describe some important issues related to the simulations, such as the network configuration for simulations, the TCP traffic flow model used in simulations and the implementation of the continuous time controller.

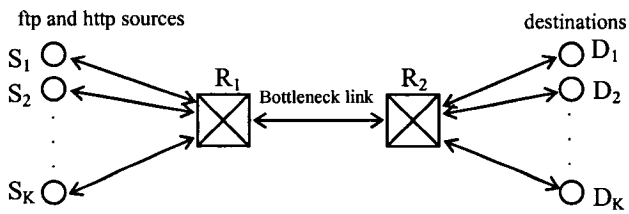

Fig. 2.3: Bottleneck link configuration

2.3.1 Network Configuration for Simulation

Throughout the thesis, the simulation model we used is the typical bottleneck configuration as shown in Fig. 2.3. Each S_i is a TCP-Reno source. D_i is a destination corresponding to S_i. R_1 and R_2 are routers. The network is configured as such that only the link between R_1 and R_2 is the bottleneck link. We simulated the case of long-lived greedy ftp connections as well as short-lived http sources. The receiver's advertised window size is set sufficiently large so that TCP connections are not constrained at the destination. As a comparison, the other active queue management algorithms, the RED [FlJa93], the P controller and the PI

controller [HoMi02], and the AVQ [KuSr04] are also implemented in our simulations, and their performance compared.

We have run different scenarios with different settings of N, R_0 and C, and we will describe them in the corresponding chapters.

2.3.2 TCP Traffic Flow Model

Two different kinds of sources are used in this thesis: long-lived flow and short-lived flow. A long-lived flow source (e.g. a ftp source) continuously sends packet out as long as the TCP congestion window allows. It accounts for the file transfer traffic: the file size is considered as infinite while the think time between any two transfers is zero. A short-lived flow source (e.g. a http source) generates the file that has a limited file size. After one file transfer completes, the source will get into an idle period for the duration of a think time. Then it starts to create and send the next file.

In the short-lived flow sources, we have to determine the file size and the think time. We shall use the Neidhardt model [Neid96] here as the mathematical model for the file size of the short-lived flow. This model gives the file size (in bytes) distribution as:

$$P\{F > f\} = (1 + (f/\alpha)^{\eta})^{-1} \quad , \tag{2.2}$$

where α is the median; $\eta > 1$; $\alpha \times \eta^{-1} \times \Gamma(1/\eta) \times \Gamma(1\text{-}1/\eta)$ is the mean. If $0 < \eta \leq 1$, Equ. (2.2) is a probability distribution without first moment. From the data collected, Neidhardt found $\alpha =$ 2190, and $\eta = 1.15066$. For the think time (in seconds) between any two transfers, we use the same model as in [OtLa99] that has a distribution of

$$P\{T > t\} = p_h(1 + (t/\alpha_h)^{\eta_h})^{-1} + p_l(1 + (t/\alpha_l)^{\eta_l})^{-1} \quad , \tag{2.3}$$

where $p_h + p_l = 1$, so (2.3) is a mixture of two distributions in (2.2), and

$$p_h = 0.4953916 \, , \; p_l = 0.5046084 \, ;$$

$$\alpha_h = 1.0 \, , \; \alpha_l = 0.0245032 \, ;$$

$$\eta_h = 1.243437 \, , \; \eta_l = 3.252665 \, .$$

Long-lived flows can be modeled by the think time as well. For example, infinitely long files used in our study are equivalent to a source with zero think time. In this case, a source will always have data to send as long as their congestion window allows.

2.3.3 Implementation of the *s*-domain Controllers

In this thesis, all the designed controllers are described in the *s*-domain (Laplace transform). For a digital implementation in the simulation, we need to convert the description into a *z*-transform. In our work we use the Tustin's approximation [AsWi84] to obtain the *z*-domain transfer function from the *s*-domain one, i.e., we use

$$s = 2f_S \frac{z-1}{z+1}, \tag{2.4}$$

where f_S is the sampling frequency. It is advisable to operate the digital controller at 10-20 times the loop bandwidth [AsWi84]. Once we get the *z*-transform of the controller, we could then implement the AQM controller in the calculation of the drop probability in every sampling period. This shall be illustrated in the following example.

Consider the following *s*-domain controller (a PI controller) $C(s) = K_P + K_I / s$. Let q_0 be the desired queue length to which we want to regulate, $p_0 = 2N^2/(R_0C)^2$ be the packet drop probability at the operating point [HoMi02], and q and p be the instantaneous queue size and packet drop probability, respectively. Using (2.4), we can translate $C(s)$ into a *z*-domain transfer function $C(z)$ between $\delta q = q - q_0$ and $\delta p = p - p_0$,

$$C(z) = \frac{dz+e}{z-1}, \tag{2.5}$$

where $d = 2K_I f_S + K_P$, $e = 2K_I f_S - K_P$. Then we have

$$\frac{\delta p(z)}{\delta q(z)} = \frac{dz+e}{z-1}. \tag{2.6}$$

Equ. (2.6) can then be converted into a difference equation of the variables yielding, at time $t = nT$, where $T = 1/f_S$,

$$\delta p(nT) = d\delta q(nT) + e\delta q((n-1)T) + \delta p((n-1)T). \tag{2.7}$$

From (2.7), we can further derive

$$p(nT) = d(q(nT) - q_0) + e(q((n-1)T) - q_0) + p((n-1)T). \tag{2.8}$$

Equ. (2.8) can be implemented in the following pseudo code, which is called at every sampling instant.

```
p      :=   d*(q-q_ref)+e*(q_old-q_ref)+p_old
p_old  :=   p
q_old  :=   q
```

2.4 Assumptions

The following assumptions pertain to the remainder of the thesis.

1) In the bottleneck configuration (Fig. 2.3), the source and the destination have one to one correspondence, i.e. S_i is the source of D_i only, i=1, 2, ...K.
2) The throughput of a TCP connection is not limited by the receiver's advertised window size because we can always make the receiver's advertised window sufficiently large.
3) The buffer size in Router 2 (R_2 in Fig. 2.3) is large enough in order to approximate no packet loss on that router.
4) There is only one congested bottleneck link for simplicity reason because we only focus on the design of AQM controllers using different control theoretical approaches, and investigates their performance under the typical single bottleneck-link configuration shown in Fig. 2.3.

Chapter 3 Pole Placement Controllers

We start our research using classical control on the study and proposal of the following two AQM controllers: the Proportional controller based on Pole Placement (the P_PP controller) and the Proportional-plus-Integral controller based on Pole Placement (the PI_PP controller). The formula derivation and the design procedure for the P_PP controller are described in Section 3.1, and for the PI_PP controller in Section 3.3. The performance evaluations for the P_PP controller and the PI_PP controller are given in Section 3.2 and Section 3.4 respectively.

3.1 Design of the P_PP Controller

The design here relies on a second order system that can be characterized by the damping ratio ξ and the undamped natural frequency ω_n. We first derive the formulae that relate ξ and ω_n to the controller parameter K_P. Based on these formulae, we then propose the design procedure for the P_PP controller and provide a design example. Appendix A summarizes the essential background knowledge on the pole placement technique. More details can be found in a standard textbook like [Ogat97].

3.1.1 Determining the P_PP Controller Parameter

The transfer function of a proportional controller is $C(s) = K_P$, where K_P is the proportional gain. Upon substitution by (2.1), the closed-loop transfer function of the system shown in Fig. 2.2 becomes

$$F(s) = \frac{K_P \frac{C^2}{2N} e^{-sR_0}}{(s + \frac{2N}{R_0^2 C})(s + \frac{1}{R_0}) + K_P \frac{C^2}{2N} e^{-sR_0}} .$$

Then the characteristic equation is

$$(s + \frac{2N}{R_0^2 C})(s + \frac{1}{R_0}) + K_P \frac{C^2}{2N} e^{-sR_0} = 0 . \qquad (3.1)$$

As will be discussed in Section 3.1.2, we assume[5]

[5] A more accurate assumption might be $|sR_0| \ll 1$, but for the control system like the TCP/AQM system, as justified by the simulations, the assumption $|sR_0| < 1$ is accurate enough for the approximation in this paper.

$$|sR_0| < 1 \ , \tag{3.2}$$

such that

$$e^{-sR_0} \approx 1 - sR_0 \ . \tag{3.3}$$

Substituting (3.3) in (3.1), the characteristic equation can be rewritten in the form

$$s^2 + as + b = 0 \ , \tag{3.4}$$

where

$$a = \frac{1}{R_0} + \frac{2N}{R_0^2 C} - \frac{C^2 R_0}{2N} K_P \ , \tag{3.5}$$

and

$$b = \frac{2N}{R_0^3 C} + \frac{C^2}{2N} K_P \ . \tag{3.6}$$

Let s_1, s_2 denote the two roots of equation (3.4), it is well known that

$$s_1 + s_2 = -a \ , \tag{3.7}$$

$$s_1 s_2 = b \ . \tag{3.8}$$

Since s_1, s_2 are also the two poles of the closed-loop system shown in Fig. 2.2, then depending on the value of the damping ratio ξ, there are 3 cases where stability can be achieved, i.e., for the poles in the left-half s-plane.

3.1.1.1 0<ξ<1

In this case, both s_1 and s_2 are complex numbers, to place the poles in the left-half s-plane, we only require $\omega_n > 0$. Let

$$s_1 = -\xi\omega_n + j\omega_n\sqrt{1-\xi^2} \ , \tag{3.9}$$

$$s_2 = -\xi\omega_n - j\omega_n\sqrt{1-\xi^2} \ . \tag{3.10}$$

Substituting (3.5), (3.9), (3.10) in (3.7), we get

$$K_P = \frac{\frac{1}{R_0} + \frac{2N}{R_0^2 C} - 2\xi\omega_n}{\frac{C^2 R_0}{2N}} \ . \tag{3.11}$$

Similarly, by substituting (3.6), (3.9), (3.10) in (3.8), we have another expression for K_P,

$$K_P = \frac{\omega_n^2 - \frac{2N}{R_0^3 C}}{\frac{C^2}{2N}} \quad . \tag{3.12}$$

By equating (3.11) to (3.12), and restricting to positive values, ω_n can be determined as

$$\omega_n = \frac{\sqrt{\xi^2 + \frac{4N}{R_0 C} + 1} - \xi}{R_0} \quad . \tag{3.13}$$

Substituting (3.13) in (3.12), equation (3.12) can be further reduced to

$$K_P = \frac{2NR_0C(2\xi^2 - 2\xi\sqrt{\xi^2 + \frac{4N}{R_0 C} + 1} + 1) + 4N^2}{R_0^3 C^3} \quad .$$

We shall use equations (3.12) and (3.13) to determine the P_PP controller parameter K_P from ξ.

3.1.1.2 ξ=1

Let $s_1 = s_2 = -\omega_n$. Through the same derivation as in Section 3.1.1.1, we get similar formulae like (3.12), (3.13) and with $\xi = 1$, we have

$$\omega_n = \frac{\sqrt{2 + \frac{4N}{R_0 C}} - 1}{R_0} \quad , \text{ and } K_P = \frac{2NR_0C(3 - 2\sqrt{2 + \frac{4N}{R_0 C}}) + 4N^2}{R_0^3 C^3} \quad .$$

3.1.1.3 ξ>1

In this case, both poles are real. Now let

$$s_1 = -\xi\omega_n + \omega_n\sqrt{\xi^2 - 1} \quad , \; s_2 = -\xi\omega_n - \omega_n\sqrt{\xi^2 - 1} \quad ,$$

going through the same arguments as in Section 3.1.1.1, the same results can be obtained, i.e. K_P is determined by (3.12), (3.13).

3.1.2 Additional Requirements from $|sR_0| < 1$

Recall that in the derivation presented in Section 3.1.1, Equ. (3.2) is an assumption that must be satisfied. This means the two poles s_1, s_2 must satisfy (3.2) as well. Actually this condition can be easily met as long as we choose appropriate ξ. We can discuss this in two cases

according to the value of ξ:

3.1.2.1 $0<\xi\leq 1$

In this case, we have $|s_1 R_0| = |s_2 R_0| = \omega_n R_0$. Therefore, to satisfy (3.2), we only need to choose suitable ξ such that

$$\omega_n R_0 < 1 \quad . \tag{3.14}$$

From (3.13) and (3.14), we can obtain the condition

$$\xi > \frac{2N}{R_0 C} \quad . \tag{3.15}$$

3.1.2.2 $\xi>1$

In this case $|sR_0|$ becomes

$$|s_1 R_0| = \omega_n R_0 (\xi - \sqrt{\xi^2 - 1}) < \omega_n R_0 < 2\xi\omega_n R_0 \quad ,$$

$$|s_2 R_0| = \omega_n R_0 (\xi + \sqrt{\xi^2 - 1}) < 2\xi\omega_n R_0 \quad .$$

This gives us the condition

$$2\xi\omega_n R_0 < 1 \quad . \tag{3.16}$$

3.1.3 The P_PP Controller Design Algorithm

From the previous description, the design procedure for the P_PP controller can start with $0<\xi\leq 1$ or $\xi>1$.

<u>$0<\xi\leq 1$</u>

1) Choose ξ from $0<\xi\leq 1$, and such that it satisfies (3.15).
2) Calculate ω_n from (3.13).
3) Calculate K_P from (3.12), the procedure ends.

<u>$\xi>1$</u>

1) Pick an arbitrary value of $\xi>1$.
2) Calculate ω_n from (3.13).
3) Check whether (3.16) is satisfied,
3a) if answer is positive, continue with step 4);
3b) otherwise go back to step 1) to adjust ξ.
4) Calculate K_P from (3.12), the procedure ends.

Note that from the control theory [Ogat97], when $0<\xi\leq 1$, lower value of ξ causes

larger overshoots but also means shorter response time; when ξ>1, there is no overshoot, but the response time becomes longer when ξ increases. Our many simulation results show that when the value of ξ is taken within the range $0.2 \leq \xi \leq 0.8$, the system has a satisfactory transient response performance. Another parameter we need to specify for the P_PP controller is the reference queue size (REF_QS), which is the desired average queue size at the bottleneck node. Since there is no integral action in the P_PP controller, an offset between the REF_QS and the steady-state average queue size does exist. This will be illustrated in the simulation results. Obviously lower REF_QS means shorter average packet delay and smaller buffer utilization. The recommended REF_QS is 10%~50% of the buffer size (see Section 3.2 Experiment 3).

3.1.4 A Design Example for the P_PP Controller

Suppose we have N=100, R_0=0.25 second, C=9708 packets/second. We choose ξ=0.2. Obviously $\xi > \frac{2N}{R_0 C}$=0.0824 satisfies (3.15). We therefore obtain ω_n=3.591 from (3.13) and K_P=2.456×10^{-5} from (3.12). This design gives satisfactory system performance as we can see from simulation results shown in Section 3.2. If we choose ξ=1.1. From (3.13) we get ω_n =1.764. Thus $2\xi\omega_n R_0$ =0.9702<1 satisfies condition (3.16). We then determines K_P=3.806×10^{-6} from (3.12). The controller is implemented in OPNET simulations using the method described in Section 2.3.3.

3.2 Performance Evaluation for the P_PP Controller

The P_PP controller has been verified via simulations using OPNET Modeler [OPNE01]. The simulation model is the typical bottleneck configuration as described in Section 2.3.1. In all the simulations described in this thesis, we have run different scenarios with different settings of N, R_0, C and the simulation length, and we will describe them in the corresponding chapters if they are different from the default setting as described below.

By default, the simulation length is set as 100 seconds; and we use 100 greedy ftp sources and 20 http sources[6]; the round trip time is set as 0.25 second; the link bandwidth is

[6] To test the robustness of the proposed controller, we introduce the http sources as a disturbance. Note that when we count N (number of connections), we only consider the long-lived ftp sources, which is assumed by the fluid flow model.

T3 (44,736,000bps), with the average packet size being 576 bytes, which means that the link capacity is 44736000/576/8 = 9708 packets/second; the buffer size is 1000 packets; therefore, we have: N=100, R_0=0.25 second, C=9708 packets/second. We use the following settings for the RED algorithm: the maximum value for drop probability max_p is 0.1; the maximum queue threshold max_{th} is 700; the minimum queue threshold min_{th} is 150; the queue weight w_q is 1.33×10^{-6}. According to the design guidelines in [KuSr04], we use the following settings for AVQ algorithm: the desired utilization γ is 0.98, and the damping factor α is 0.15.

In the simulations of this section, we use the default settings as described in the previous paragraph, and the REF_QS[7] is set to 150 packets except in Experiment 3. In all the graphs shown in this section, we depict the time evolution of the instantaneous queue size or the average queue size, with the time axis drawn in seconds. The notation K means the *damping ratio* ξ, and REF_QS means *reference queue size.*

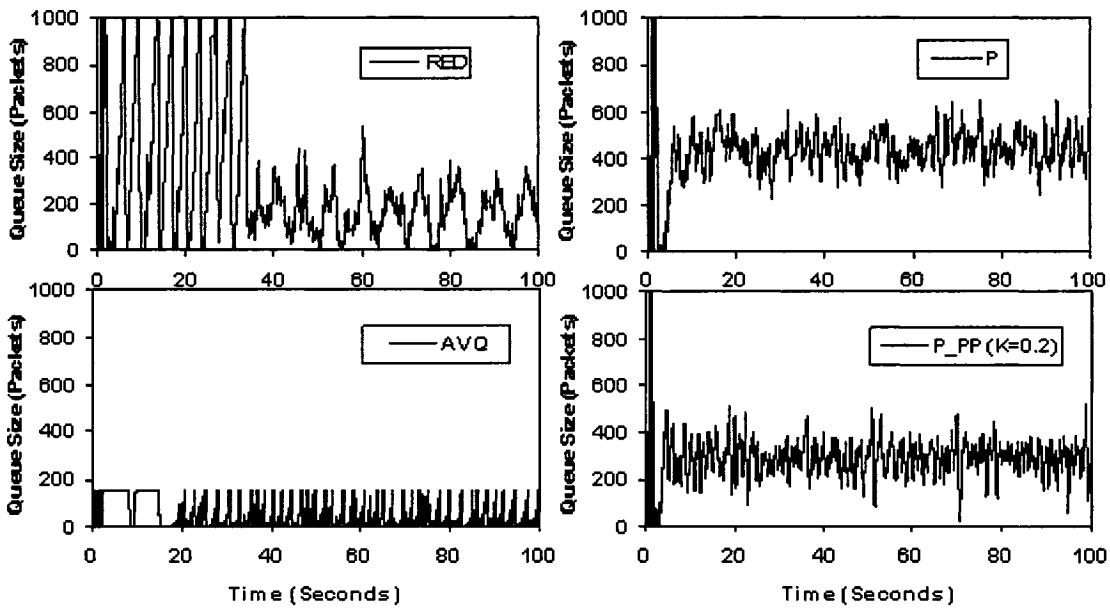

Fig. 3.1: Instantaneous queue size vs. Time: comparison of the RED, the P, the AVQ and the P_PP controllers (K=ξ=0.2)

[7] Since AVQ does not directly attempt to control the queue size, thus an experiment in [KuSr04] has adopted dropping every packet that arrives when the real queue reaches the desired queue length. Therefore, we also drop every packet that arrives when the real queue reaches the drop-threshold parameter that is equivalent to the REF_QS in our algorithm.

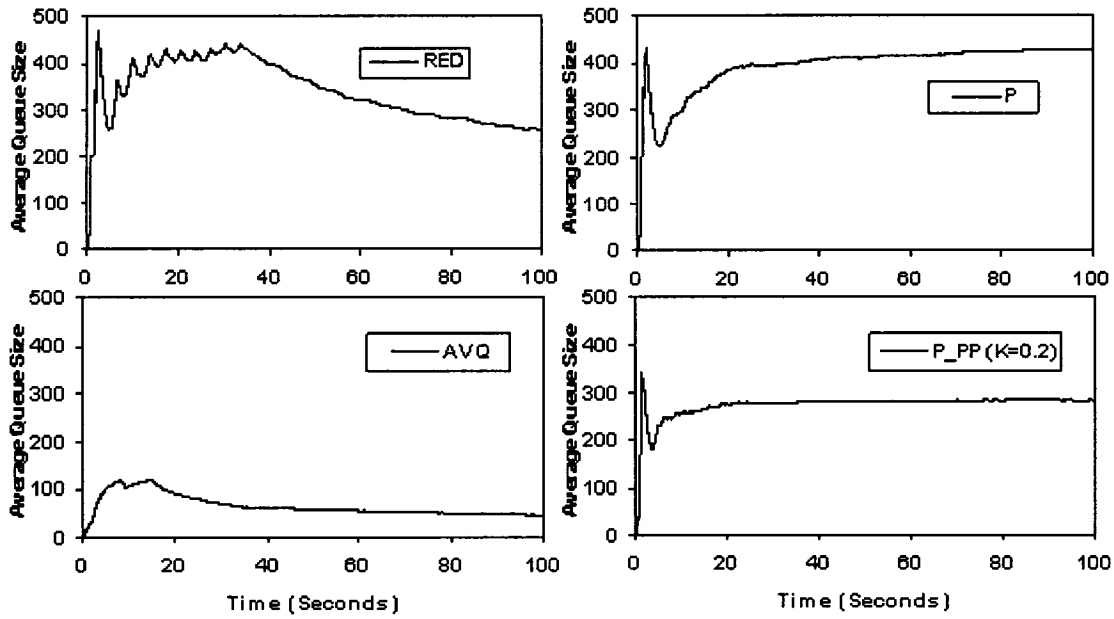

Fig. 3.2: Average queue size vs. Time: comparison of the RED, the P, the AVQ and the P_PP controllers (K=ξ=0.2)

Experiment 1 - Comparison of RED, P, AVQ and P_PP

This experiment compares the performance of the RED [FlJa93], the P controller [HoMi02], the AVQ [KuSr04] and the P_PP controller. Choosing ξ=0.2, and using the procedure presented in Section 3.1.3, we could get K_P=2.456×10^{-5} for the P_PP controller.

Fig. 3.1 and Fig. 3.2 present the instantaneous queue size and the average queue size respectively. From Fig. 3.2, the settling time[8] of the RED algorithm is greater than 100 seconds, while the settling times for the P controller, the AVQ algorithm and the P_PP controller are 50 seconds, 40 seconds and 20 seconds respectively. Namely the RED has the longest settling time, with the P controller the second longest, the AVQ the third longest, while the P_PP controller has the shortest among the four. Fig. 3.1 justifies the conclusion that the queue size oscillations of the RED are larger than those of the P controller, the AVQ and the P_PP controller. It is also shown that the steady-state queue size of the P controller (420 packets) is larger than that of the P_PP controller (280 packets), which in turn is larger than that of the AVQ (50 packets). Since the REF_QS is set at 150 packets, this means the offsets are 270, 130 and 100 packets for the P controller, the P_PP controller and the AVQ

[8] The settling time is defined as the time required for the curve to reach and stay within ±5% around the final value [Ogat97].

respectively.

Table 3.1 Design parameters with different ξ

ξ	ω_n	K_P	$\omega_n R_0$	$2\xi\omega_n R_0$
0.2	3.591	2.456×10^{-5}	0.898	-----
0.6	2.539	1.089×10^{-5}	0.635	-----
1.1	1.764	3.806×10^{-6}	-----	0.970

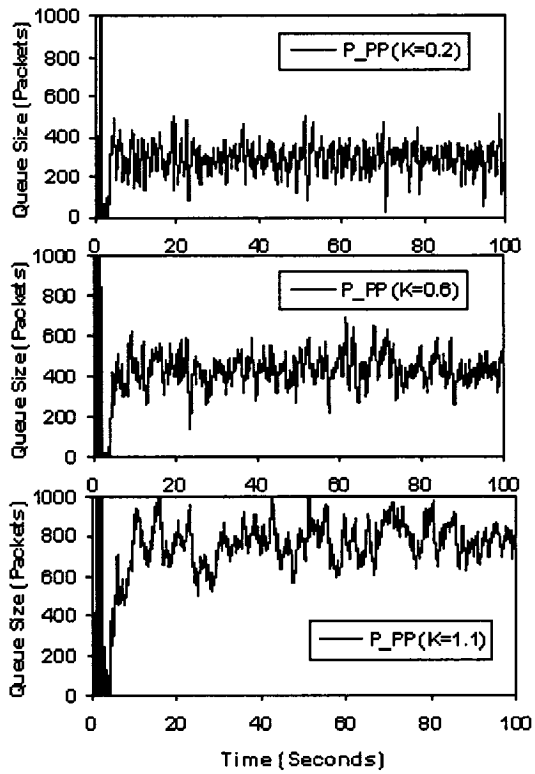

Fig. 3.3: Instantaneous queue size vs. Time: comparison of the P_PP controller with different damping ratios K (K=ξ=0.2, 0.6, 1.1)

Experiment 2 - Investigating the impact of ξ

In this experiment, we adjust ξ to different values in order to investigate its impact. Table 3.1 gives the design parameters ω_n and K_P for different ξ using the algorithm presented in Section 3.1.3. Fig. 3.3 shows the instantaneous queue size. Although the fluctuations appear to be comparable, the mean queue size is an increasing function of ξ (K). This is also confirmed in Fig. 3.4 where one can compare the steady-state mean queue values. In Fig. 3.4, we deliberately use different scales for the vertical axis to show the overshoot clearly. By "overshoot", we mean the initial peak value is greater than the steady-state value. When ξ<1 (ξ=0.2 and 0.6), the average queue size curves show overshoots. Note that the overshoot gets smaller when ξ increases. However, when ξ>1 (ξ=1.1), the curve shows no overshoot at all.

These conclusions conform to the control theoretical analysis [Ogat97]. We could also conclude from Fig. 3.3 and Fig. 3.4 that the settling time increases as ξ becomes larger and the steady-state queue size changes from 280 packets, 420 packets to 740 packets as ξ varies from 0.2, 0.6 to 1.1, respectively. In other words, the steady-state offsets are 130 packets, 270 packets and 590 correspondingly.

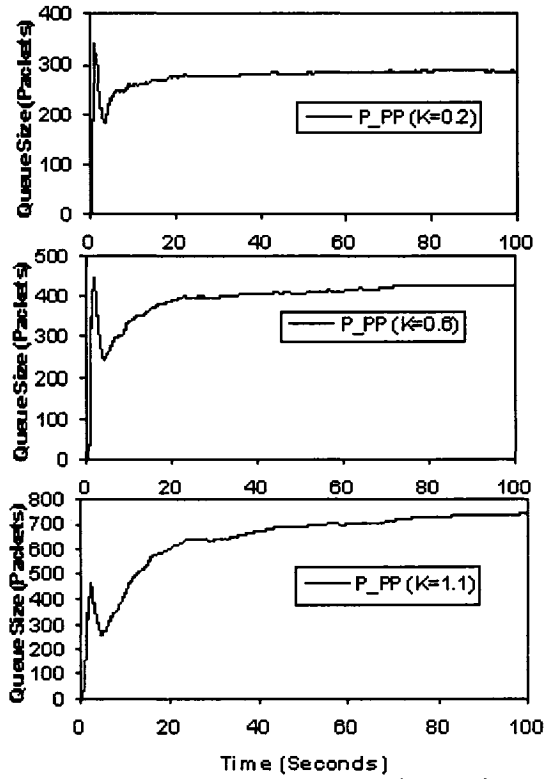

Fig. 3.4: Average queue size vs. Time: comparison of the P_PP controller with different damping ratios K (K=ξ=0.2, 0.6, 1.1)

Experiment 3 - Different REF_QS

Now we investigate the case when the REF_QS (reference queue size) is set to different values. Fig. 3.5 gives the instantaneous queue size when REF_QS=100, 300, 500 packets. It is shown that the P_PP controller cannot clamp the queue size at the reference value. The average queue size converges to another steady-state value, with an offset from the REF_QS. The steady-state average queue size is 240, 410 or 590 packets that correspond to the value of 100, 300 or 500 packets for REF_QS respectively. This means the steady-state offset is 140, 110 or 90 packets correspondingly. We know that larger REF_QS means longer average packet delay but higher buffer utilization. To seek a tradeoff between delay and utilization, and considering there is an offset between the REF_QS and the actual steady-state queue

size, the recommended REF_QS for the P_PP controller is 10%~50% of the buffer size.

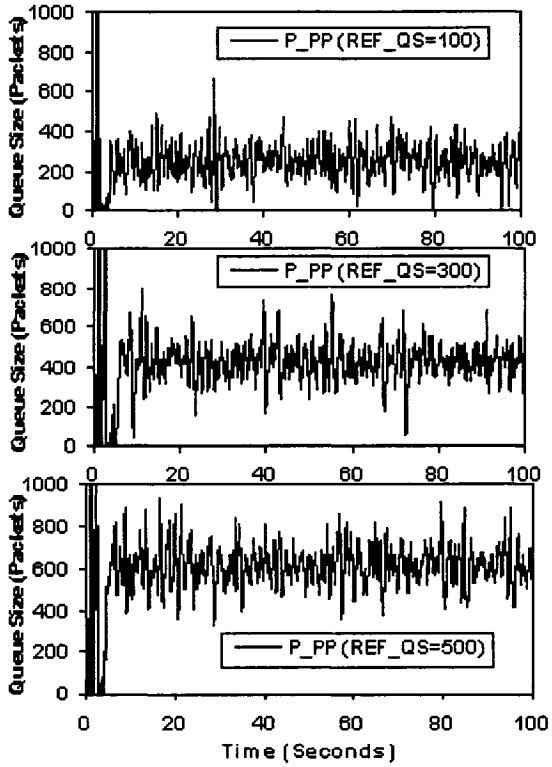

Fig. 3.5: Instantaneous queue size evolution: comparison of the P_PP controller with different REF_QS (REF_QS=100, 300, 500, ξ=0.2)

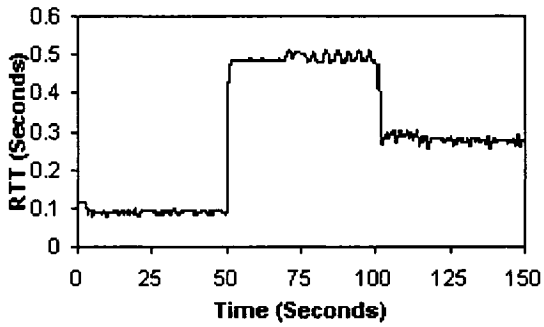

Fig. 3.6: Round trip time (RTT) evolution for the P_PP Controller in Experiment 4

Experiment 4 - Time-varying RTT

In order to investigate the P_PP controller performance when the RTT (round trip time) is time varying, we allow the RTPDs (round trip propagation delays) of all connections to change w.r.t. time: 20ms (0~50 sec), 480ms (50~100 sec) and 250ms (100~150 sec). The varying nature is reflected in Fig. 3.6 when RTT is equal to the RTPD plus the queuing delay.

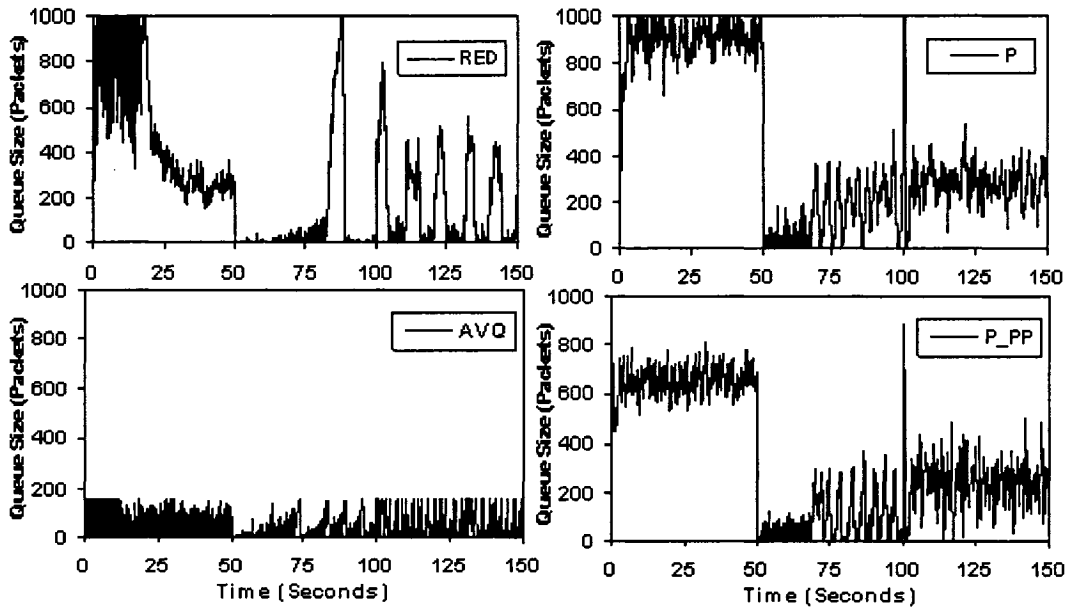

Fig. 3.7: Instantaneous queue size vs. Time: comparison of the RED, the P, the AVQ and the P_PP controllers (ξ=0.2, REF_QS=150) in the case of time-varying RTT

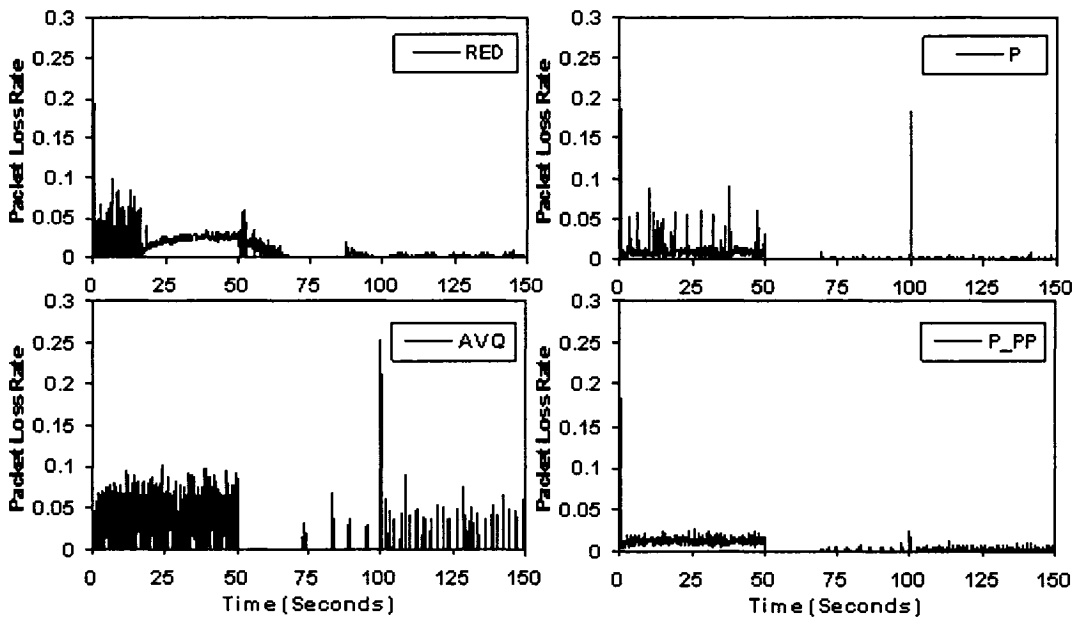

Fig. 3.8: Packet loss rate vs. Time: comparison of the RED, the P, the AVQ and the P_PP controllers (ξ=0.2, REF_QS=150) in the case of time-varying RTT

Fig. 3.7 gives the instantaneous queue size for the RED, the P controller, the AVQ and the P_PP controller subject to the above time-varying influence. In the period 0~50 sec, the P controller and the P_PP controller can quickly stabilize the queue size, while the RED takes

about 30 seconds to stabilize it. In the period 50~100 sec and the period 100~150 sec, both the P controller and the P_PP controller can still stabilize the queue size and they shows almost the same settling time, whereas the RED becomes very difficult to stabilize the queue size. In all the three simulation periods, the average queue size under the P controller is larger than that under the P_PP controller. For the AVQ algorithm, it can enforce (by dropping packets if necessary) the queue size to a low average value in all the three simulation periods, but it shows a large packet loss rate[9] in the first simulation period of 0~50 sec, as illustrated later in Fig. 3.8.

Fig. 3.8 provides the packet loss rate under the four controllers. At the beginning of the simulation ($t=0^+$), the RED, the P controller and the P_PP controller show a peak value of 17%~19%, whereas the AVQ presents a smaller one (5%). In the simulation period of 0 to 50 sec after the peak, the AVQ shows much more frequently high loss rate values (at around 8%) while the RED and the P controller vary from 0 to 10%. However, our P_PP controller gives much less loss rate that varies between 0%~2%. In the second simulation period of 50 to 100 sec, the loss rates of the RED and the AVQ controllers vary within the range 0%~5%, while the P controller and the P_PP controller vary between 0%~1% only. During the third simulation period of 100 to 150 sec, the RED, the P controller and the P_PP controller give similar loss rate variations, within the range 0%~1%, but the AVQ shows larger loss rate variations within the range 0%~10%.

So far we have simulated the AVQ algorithm according to the drop-threshold (REF_QS) used in the experiment of [KuSr04]. However, we found its loss rate is much higher than the P_PP while not achieving the desired average queue length. In order to have a fair comparison, we choose to simulate the case that the drop-threshold is set to 2 times REF_QS, which gives us the value of 300 packets for the drop-threshold (hoping the loss rate can be reduced). We use the same network scenario as stated in the beginning of Experiment 4. The instantaneous queue size evolution and the packet loss rate evolution are shown in Fig. 3.9 and Fig. 3.10 respectively. Unfortunately we do not find much improvement. For example, in the third simulation period of 100 sec to 150 sec in Fig. 3.9, the instantaneous queue size shows frequent large oscillations between 0 and 300 packets

[9] Packet loss rate is defined as the ratio of the total number of packets dropped by the router divided by the total number of packets entering the router within a specific period of time.

(compare to the case when the AVQ drop-threshold is only REF_QS), meaning that the queue delay oscillations would also be large. Likewise, when comparing Fig. 3.10 with Fig. 3.8, we can also conclude that the packet loss rates of the AVQ algorithm are larger in the case of setting drop-threshold to 2 times REF_QS.

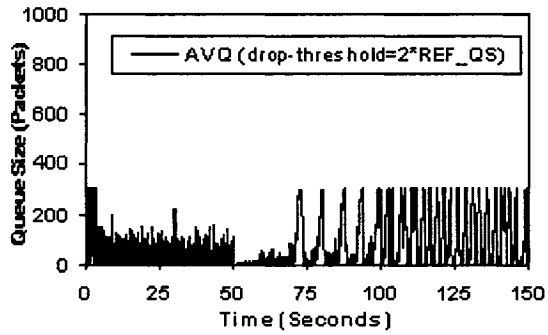

Fig. 3.9: Instantaneous queue size vs. Time: the AVQ with the drop-threshold set to 2×REF_QS = 300 packets in the case of time-varying RTT

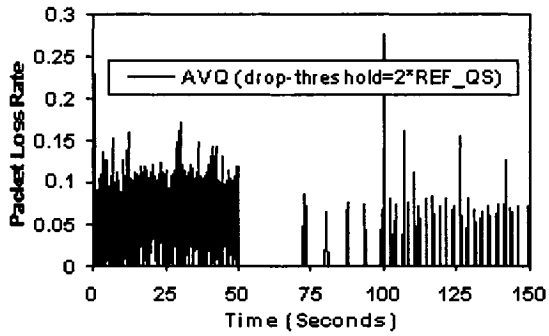

Fig. 3.10: Packet loss rate vs. Time: the AVQ with the drop-threshold set to 2×REF_QS = 300 packets in the case of time-varying RTT

Other experiments in addition to the above show that increasing the AVQ drop-threshold in general causes larger queue size oscillations and worse packet loss rates. Since the larger drop-threshold is not helping, we revert back to what the author [KuSr04] had proposed in his experiment, namely, setting the drop-threshold to the REF_QS for all later comparisons.

Experiment 5 - Different RTT for different connections

We have in total 100 ftp connections and 20 http connections, and each can have different RTT. For the ftp sources, the RTPDs are 100ms, 200ms, 300ms and 400ms for the connection 1~25, connection 26~50, connection 51~75 and connection 76~100 respectively,

and for the http sources, the RTPDs are set to 100ms and 400ms for connection 1~10 and connection 11~20 respectively.

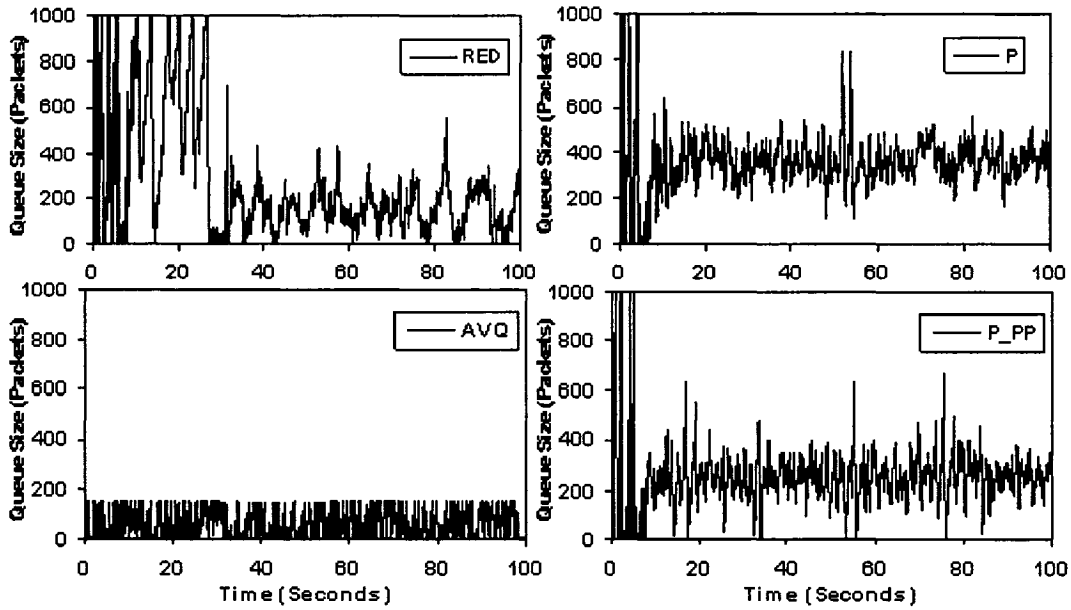

Fig. 3.11: Instantaneous queue size vs. Time: comparison of the RED, the P, the AVQ and the P_PP controllers (ξ=0.2, REF_QS=150) in the case of different RTTs for different connections

The instantaneous queue size is given in Fig. 3.11. It is shown that all the four controllers can stabilize the queue size, but with different settling times. The RED takes 30 seconds to stabilize the queues size, whereas the P controller and the P_PP controller only takes 7 seconds to clamp the queue size to the steady state, and the AVQ can drive the queue size to a lower steady-state value from the very beginning. Lower average queue size means shorter average queuing delay, but it also means lower buffer utilization. The steady-state average queue sizes for the four controllers are also different: 150 packets (RED), 370 packets (P), 75 packets (AVQ) and 270 packets (P_PP).

Table 3.2 Round trip propagation delay (RTPD) configuration for Experiment 6

Simulation time t (sec)	$0 < t \leq 50$	$50 < t \leq 100$	$100 < t \leq 150$
Connection ID	RTPD (ms)	RTPD (ms)	RTPD (ms)
ftp 1~25	100	200	150
ftp 26~50	200	400	300
ftp 51~75	300	600	450
ftp 76~100	400	800	600
http 1~10	100	200	150
http 11~20	400	800	600

Experiment 6 - Different time-varying RTT for different connections

In this experiment, we further investigate the case in which each connection has its RTPD change w.r.t. time in addition to having different RTPD from each other. Table 3.2 summarizes the changing values.

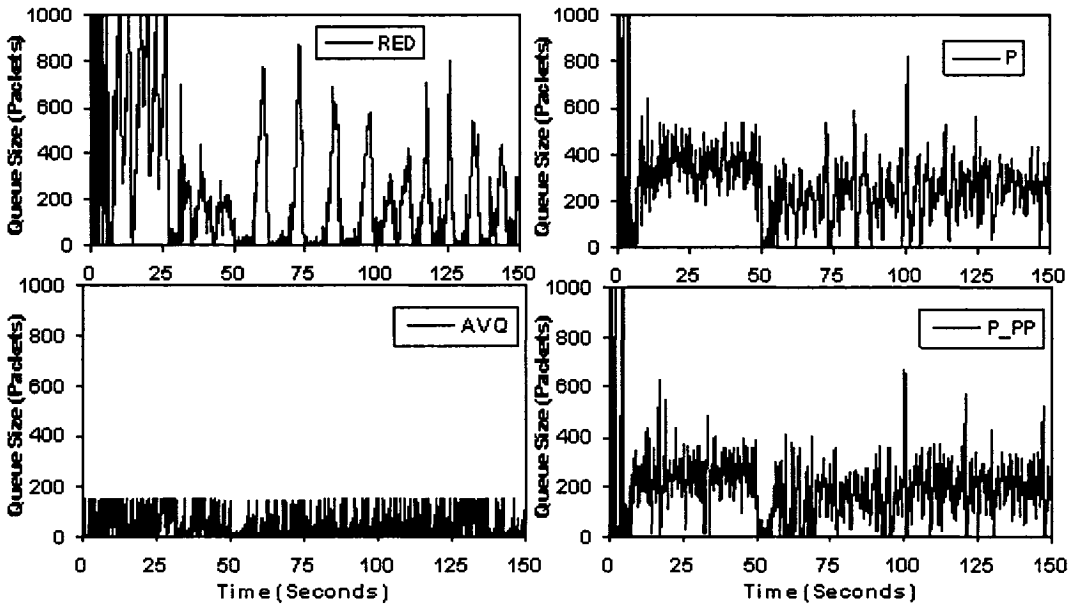

Fig. 3.12: Instantaneous queue size vs. Time: comparison of the RED, the P, the AVQ and the P_PP controllers in the case of different time-varying RTTs for different connections

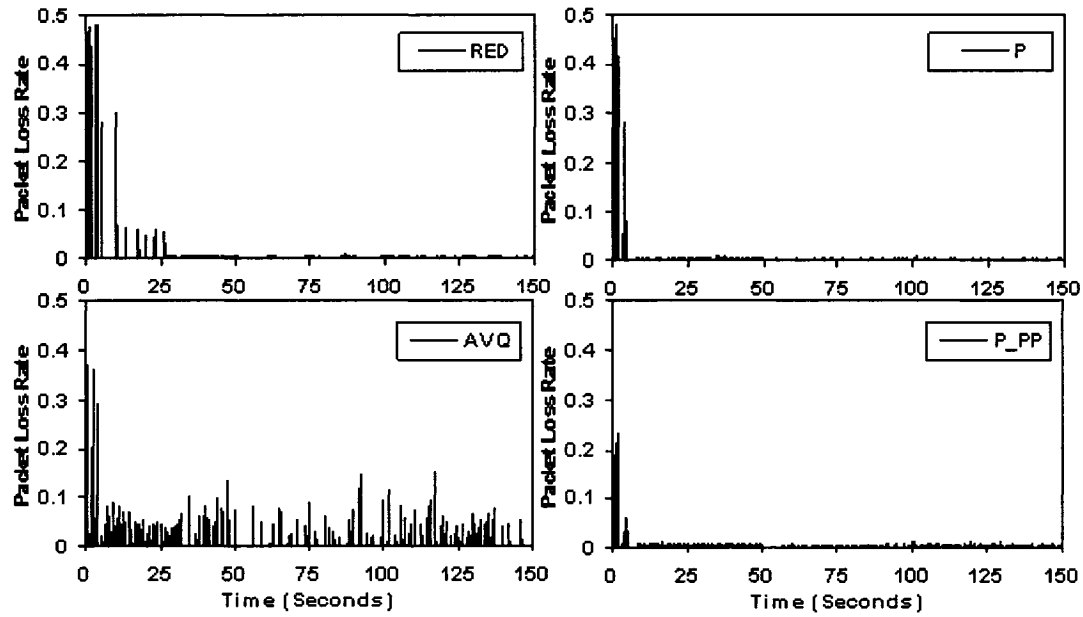

Fig. 3.13: Packet loss rate vs. Time: comparison of the RED, the P, the AVQ and the P_PP controllers in the case of different time-varying RTTs for different connections

Fig. 3.12 shows the instantaneous queue size for the four controllers. In the period 0~50 sec, it takes the RED 30 seconds to stabilize the queue size, whereas the P controller and the P_PP controller only take 7 seconds to stabilize it, and the AVQ can drive the queue size to a low value easily from the very beginning, but at the cost of larger packet loss rate, which can be inferred from Fig. 3.13 that shows the packet loss rate comparison for the four controllers. In the period 50~100 sec and the period 100~150 sec, the P controller, the AVQ and the P_PP controller can easily drive the queue size to the steady state, but the RED has difficulties to do this. During the whole simulation period, the average queue size under the P controller is larger than that under the P_PP controller.

Fig. 3.13 gives the packet loss rate comparison. At the beginning of the simulation ($t=0^+$), the RED, the P controller and the AVQ show a peak value of 38%~48%, whereas the P_PP controller presents a smaller one (22%). From 25 sec until the end of simulation, the RED, the P controller and the P_PP controller have similar loss rate variations, within the range 0%~1%. But the AVQ shows larger loss rate variations within the range 0%~15%.

3.3 Design of the PI_PP Controller

Although the P_PP controller is shown to outperform the RED and the P controller, we identify three limitations of the P_PP controller. Firstly, condition (3.15) limits the range of ξ. If R_0C is small and N is large enough, ξ may have to be greater than 1, which may result in sluggish response. Secondly, ξ and ω_n are coupled as depicted by (3.13). Thus we cannot designate them separately. The third limitation is that, under the P_PP controller, there does exist an offset (error) between the REF_QS and the actual steady-state average queue size. To overcome these limitations, we shall propose the PI_PP controller in this section as an alternative.

3.3.1 Determining the PI_PP Controller Parameters

As discussed in Section 2.2, the transfer function of a PI controller is given by $C(s) = K_P + K_I / s$, where K_P is the proportional gain, and K_I is the integral gain. Upon substitution by (2.1), the closed-loop transfer function of the system shown in Fig. 2.2 becomes

$$F(s)=\frac{(K_P s+K_I)e^{-sR_0}\dfrac{C^2}{2N}}{s(s+\dfrac{2N}{R_0^2C})(s+\dfrac{1}{R_0})+(K_P s+K_I)e^{-sR_0}\dfrac{C^2}{2N}} \quad .$$

To determine K_P and K_I, we make use of the characteristic equation

$$s(s+\frac{2N}{R_0^2C})(s+\frac{1}{R_0})+(K_P s+K_I)e^{-sR_0}\frac{C^2}{2N}=0 \quad . \tag{3.17}$$

As will be discussed in Section 3.3.2, we assume

$$|sR_0|<1 \quad , \tag{3.18}$$

such that

$$e^{-sR_0}\approx 1-sR_0 \quad . \tag{3.19}$$

Substituting (3.19) in (3.17), the characteristic equation can be rewritten in the form

$$s^3+as^2+bs+c=0 \quad , \tag{3.20}$$

where

$$a=\frac{1}{R_0}+\frac{2N}{R_0^2C}-\frac{C^2R_0}{2N}K_P \quad ; \tag{3.21}$$

$$b=\frac{2N}{R_0^3C}+\frac{C^2}{2N}K_P-\frac{C^2R_0}{2N}K_I \quad ; \tag{3.22}$$

$$c=\frac{C^2}{2N}K_I \quad . \tag{3.23}$$

Let s_1, s_2 and s_3 denote the three roots of equation (3.20), namely s_1, s_2 and s_3 are the three poles of the closed-loop system shown in Fig. 2.2. It is well known that

$$s_1+s_2+s_3=-a \quad ; \tag{3.24}$$

$$s_1s_2+s_2s_3+s_1s_3=b \quad ; \tag{3.25}$$

$$s_1s_2s_3=-c \quad . \tag{3.26}$$

Depending on the value of the damping ratio ξ, there are 3 possible ranges of ξ as discussed in the following subsections.

3.3.1.1 $0<\xi<1$

In this case, the three poles are a pair of complex conjugate poles and a real pole, i.e.

$$s_1 = -\xi\omega_n + j\omega_n\sqrt{1-\xi^2} \quad , \tag{3.27}$$

$$s_2 = -\xi\omega_n - j\omega_n\sqrt{1-\xi^2} \quad , \tag{3.28}$$

$$s_3 = -m\xi\omega_n \quad , \tag{3.29}$$

where the real number m determines the position of the third root. To ensure the stability of the system, we need $m>0$ and $\omega_n>0$. More discussions on the value m are given in Section 3.3.2. Substituting (3.21), (3.27), (3.28), (3.29) in (3.24), we have

$$K_P = \frac{\frac{1}{R_0} + \frac{2N}{R_0^2 C} - (2+m)\xi\omega_n}{\frac{C^2 R_0}{2N}} \quad . \tag{3.30}$$

Substituting (3.22), (3.27), (3.28), (3.29) in (3.25), we get

$$K_I = \frac{\frac{4N}{R_0^3 C} + \frac{1}{R_0^2} - \frac{(2+m)\xi\omega_n}{R_0} - \omega_n^2 - 2m\xi^2\omega_n^2}{\frac{C^2 R_0}{2N}} \quad . \tag{3.31}$$

Substituting (3.23), (3.27), (3.28), (3.29) in (3.26), we can derive

$$K_I = \frac{2Nm\xi\omega_n^3}{C^2} \quad . \tag{3.32}$$

By equating (3.31) to (3.32), m can be determined as

$$m = \frac{\frac{4N}{R_0^3 C} + \frac{1}{R_0^2} - \frac{2\xi\omega_n}{R_0} - \omega_n^2}{\frac{\xi\omega_n}{R_0} + 2\xi^2\omega_n^2 + \xi\omega_n^3 R_0} \quad . \tag{3.33}$$

We shall use equations (3.30), (3.32), (3.33) to determine the PI_PP controller parameters K_P and K_I from ξ and ω_n.

3.3.1.2 $\xi=1$

Let $s_1 = s_2 = -\omega_n$, $s_3 = -m\omega_n$. Through the same derivation as in Section 3.3.1.1, we get similar formula like (3.30), (3.32) and (3.33) with $\xi = 1$, we have

$$m=\frac{\frac{4N}{R_0^3C}+\frac{1}{R_0^2}-\frac{2\omega_n}{R_0}-\omega_n^2}{\frac{\omega_n}{R_0}+2\omega_n^2+\omega_n^3R_0}\quad,\quad K_P=\frac{\frac{1}{R_0}+\frac{2N}{R_0^2C}-(2+m)\omega_n}{\frac{C^2R_0}{2N}}\quad\text{and}\quad K_I=\frac{2Nm\omega_n^3}{C^2}\quad.$$

3.3.1.3 $\xi>1$

In this case, all the poles are real. Let

$$s_1=-\xi\omega_n+\omega_n\sqrt{\xi^2-1}\quad,\ s_2=-\xi\omega_n-\omega_n\sqrt{\xi^2-1}\quad,\text{and}\quad s_3=-m\xi\omega_n\quad.$$

Then by going through the same argument and derivation as in Section 3.3.1.1, the same results can be obtained, i.e. K_P and K_I can be determined by (3.30), (3.32) and (3.33).

3.3.2 Additional Requirements from $|sR_0|<1$ and the Value *m*

Note that there are three poles (s_1, s_2 and s_3) to be placed in the s-plane when we try to decide on the parameters K_P and K_I. A simple inspection on the real parts of the poles in (3.27) to (3.29) concludes that, if we require s_1 and s_2 to be dominant poles, we need $m>>1$. Actually our simulation results show that as long as we let $m>1$, the system shows satisfactory performance. Therefore, in the PI_PP controller design, we only require

$$m>1\quad. \tag{3.34}$$

Recall that in the previous derivation, (3.18) is an assumption that must be satisfied. This means all three poles s_1, s_2 and s_3 must satisfy (3.18) as well. Actually this condition can be easily met as long as we choose appropriate ξ and ω_n. We can discuss this in two cases according to the value of ξ.

3.3.2.1 $0<\xi\leq1$

In this case, $|s_1R_0|=|s_2R_0|=\omega_nR_0$; $|s_3R_0|=m\xi\omega_nR_0$. Therefore, to satisfy (3.18), we only need to choose suitable ξ and ω_n such that

$$\omega_nR_0<1\ ; \tag{3.35}$$

and

$$m\xi\omega_nR_0<1\quad. \tag{3.36}$$

3.3.2.2 $\xi>1$

In this case, $|sR_0|$ becomes

$$|s_1R_0|=\omega_nR_0(\xi-\sqrt{\xi^2-1})<\omega_nR_0<2\xi\omega_nR_0\ ;$$

$$|s_2 R_0| = \omega_n R_0 (\xi + \sqrt{\xi^2 - 1}) < 2\xi\omega_n R_0 \ ;$$

$$|s_3 R_0| = m\xi\omega_n R_0 \ .$$

Thus, as long as we choose ξ and ω_n to meet

$$2\xi\omega_n R_0 < 1 \ , \tag{3.37}$$

and

$$m\xi\omega_n R_0 < 1 \ , \tag{3.38}$$

condition (3.18) can be satisfied.

3.3.3 The PI_PP Controller Design Algorithm

From the previous description, the design procedure for the PI_PP controller can start with $0<\xi\leq 1$ or $\xi>1$.

$0<\xi\leq 1$

1) Pick an arbitrary value for ξ from $0<\xi\leq 1$, and use (3.35) to determine ω_n.
2) Calculate m from (3.33), check whether m satisfies (3.34) and (3.36),
2a) if answer is positive, continue step 3),
2b) otherwise go back to step 1) to adjust ξ and ω_n.
3) Calculate K_P from (3.30).
4) Calculate K_I from (3.32), procedure ends.

$\xi>1$

1) Choose ξ and ω_n such that (3.37) and $\xi>1$ are satisfied.
2) Calculate m from (3.33), check whether m satisfies (3.34) and (3.38),
2a) if answer is positive, continue with step 3),
2b) otherwise go back to step 1) to adjust ξ and ω_n.
3) Calculate K_P from (3.30).
4) Calculate K_I from (3.32), procedure ends.

For the same reason as presented in Section 3.1.3, the recommended value for ξ is within the range $0.2\leq\xi\leq 0.8$. From the control theory [Ogat97] we know that for a fixed value of ξ, the settling time decreases as ω_n increases. But if ω_n is too large, the third root may become dominant and this would result in unsatisfactory system performance. From our many simulation results, the recommended value for ω_n is within the range $1.0\leq\omega_n\leq 2.6$. As in the P_PP controller, we also need to specify the reference queue size (REF_QS) for the PI_PP controller. Since the PI_PP controller introduces the integral action, the steady-state

error is eliminated. The recommended REF_QS is 20%~60% of the buffer size (see Section 3.4 Experiment 4).

3.3.4 A Design Example for the PI_PP Controller

Suppose we have N=100, R_0=0.25 second, C=9708 packets/second. Choose ξ=0.5, ω_n=1.8 rad/second, obviously $\omega_n R_0$=0.45<1 satisfies (3.35). From (3.33) we obtain m=1.38>1, $m\xi\omega_n R_0$=0.31<1, therefore (3.34) and (3.36) are satisfied. From (3.30) and (3.32) we have K_P=1.095×10^{-5} and K_I=8.526×10^{-6}. This design gives satisfactory system performance as we can see from simulation results in Section 3.4.

3.4 Performance Evaluation for the PI_PP Controller

The PI_PP controller has also been verified via OPNET simulations. The simulation setup is similar to that in Section 3.2, except that, a) for the PI_PP controller we set REF_QS to 200 packets by default, other values of REF_QS may be used when necessary; b) the simulation length may be different, and these differences will be described in particular experiments. In all the graphs shown subsequently in this section, the notation OM denotes *undamped natural frequency* ω_n. In the following experiments, one should see how ξ can be varied independent of ω_n, and vice versa.

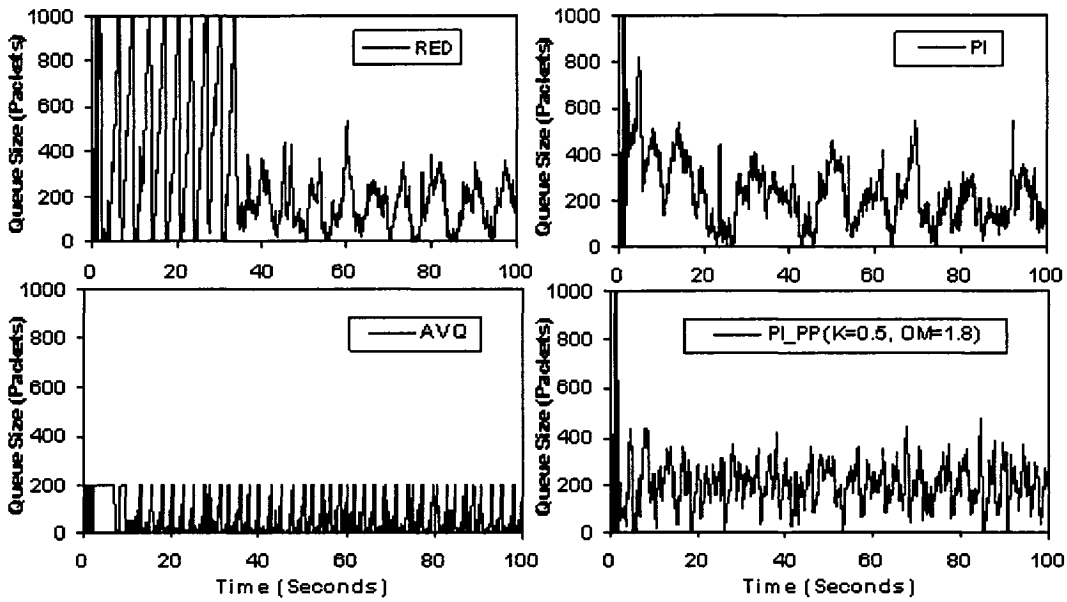

Fig. 3.14: Instantaneous queue size vs. Time: comparison of the RED, the PI, the AVQ and the PI_PP controllers (K=ξ=0.5, OM=ω_n =1.8, REF_QS=200)

Experiment 1 - Comparison of RED, PI, AVQ and PI_PP

This experiment compares the performance of the RED, the PI controller, the AVQ and the PI_PP controller. Choose ξ=0.5, ω_n =1.8, and use the algorithm in Section 3.3.3, we have K_P=1.095×10^{-5}, K_I=8.526×10^{-6}.

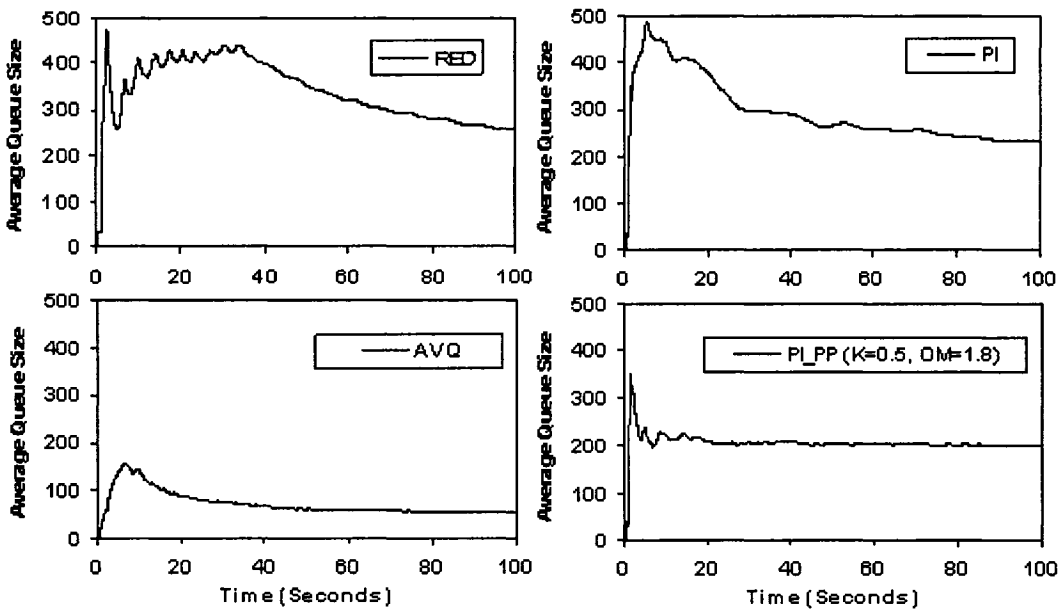

Fig. 3.15: Average queue size vs. Time: comparison of the RED, the PI, the AVQ and the PI_PP controllers (K=ξ=0.5, OM=ω_n =1.8, REF_QS=200)

Fig. 3.14 and Fig. 3.15 show the instantaneous queue size and average queue size respectively. From Fig. 3.15, the settling time of the RED algorithm is greater than 100 seconds, while the settling times for the PI controller, the AVQ and the PI_PP controller are 80 seconds, 40 seconds and 20 seconds respectively. Namely the figures show that the settling times of the four algorithms are in the order: RED > PI > AVQ > PI_PP. In addition, eventually both the PI controller and the PI_PP controller can clamp the steady-state average queue size to the reference value of 200 packets, whereas the RED and the AVQ do not have this capability. Fig. 3.14 also justifies the conclusion that the queue size oscillations of the RED and the PI controller are larger than that of the PI_PP controller and the AVQ.

Table 3.3 Design parameters with fixed ξ=0.5

ω_n	K_P	K_I	m	$m\xi\omega_n R_0$
0.4	1.091×10^{-6}	1.291×10^{-6}	19.0	0.95
1.0	6.214×10^{-6}	5.512×10^{-6}	5.20	0.65
1.8	1.095×10^{-5}	8.526×10^{-6}	1.38	0.31

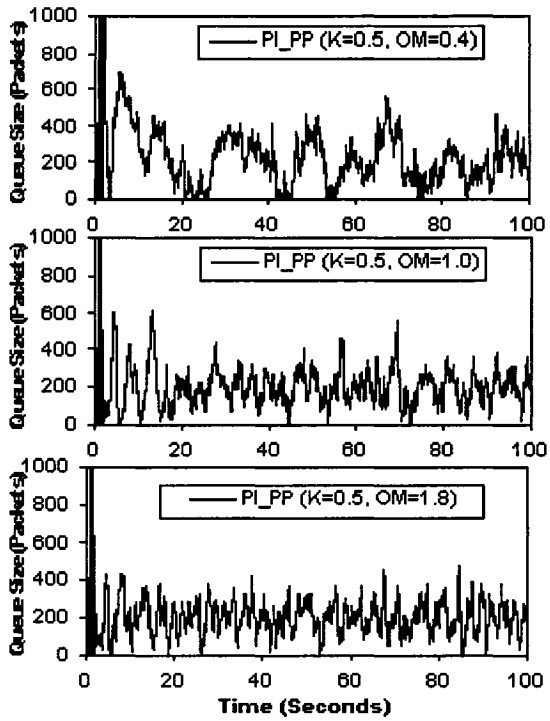

Fig. 3.16: Instantaneous queue size under the PI_PP controller with different undamped natural frequencies OM (K=ξ=0.5, REF_QS=200, OM=ω_n =0.4, 1.0, 1.8)

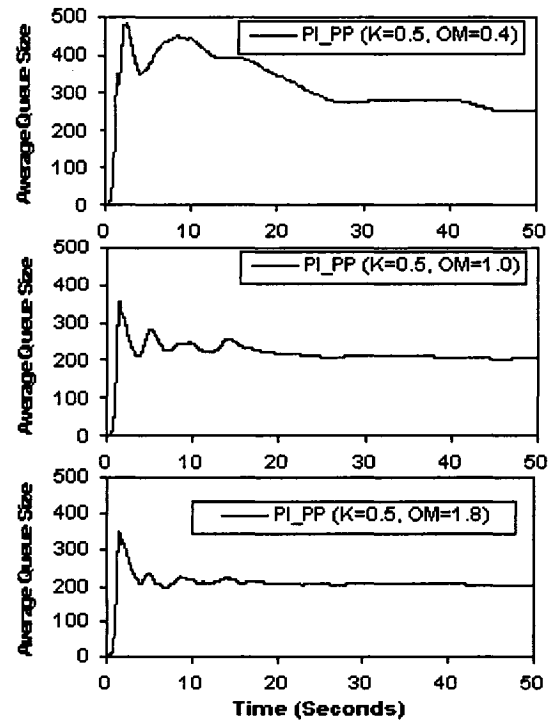

Fig. 3.17: Average queue size under the PI_PP controller with different undamped natural frequencies OM (K=ξ=0.5, REF_QS=200, OM=ω_n =0.4, 1.0, 1.8)

Experiment 2 - Investigating the impact of ω_n

In this experiment, we fix the value of ξ at 0.5 and vary ω_n to investigate its impact. Table 3.3 gives the design parameters K_P and K_I for different ω_n.

The instantaneous queue size and average queue size are shown in Fig. 3.16 and Fig. 3.17 respectively. From Fig. 3.17, the settling time for OM=ω_n=0.4 is greater than 50 seconds, while the settling times for OM=ω_n=1.0 and 1.8 are 20 seconds and 10 seconds respectively. In Appendix A, it is stated that the settling time decreases as ω_n increases for a fixed value of ξ. This is justified by the results shown in Fig. 3.16 and Fig. 3.17.

Table 3.4 Design parameters with fixed ω_n =1

ξ	K_P	K_I	m	$m\xi\omega_n R_0$
0.2	4.081×10^{-6}	7.318×10^{-6}	17.2	0.86
0.7	6.783×10^{-6}	4.521×10^{-6}	3.04	0.53
2.0	1.114×10^{-6}	4.210×10^{-7}	0.10	0.05

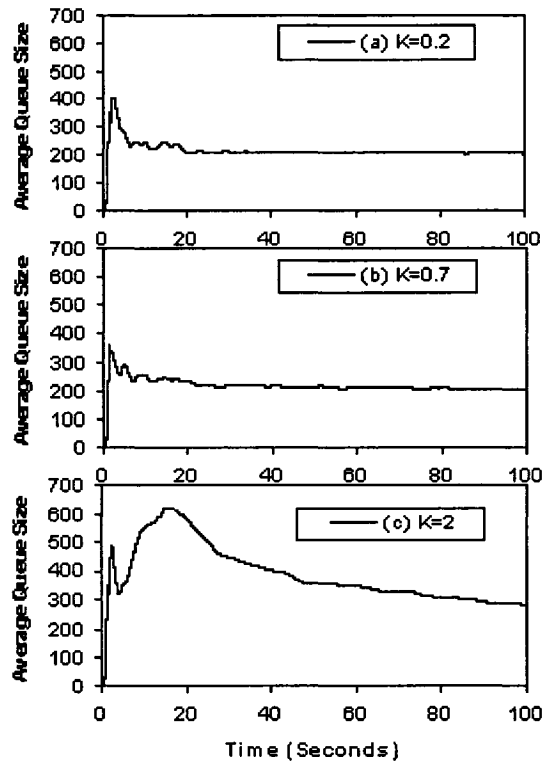

Fig. 3.18: Average queue size under the PI_PP controller with different damping ratios K (OM=ω_n =1, REF_QS=200, K=ξ=0.2, 0.7, 2)

Experiment 3 - Investigating the impact of ξ

Next we investigate the impact of ξ while fixing ω_n value at 1. Table 3.4 presents the design parameters K_P and K_I for different ξ. Note that in order to investigate the performance when the third root becomes dominant (i.e. $m<1$), we deliberately choose $\xi=2$ so that $m=0.1<1$. Meanwhile we must make sure $m>0$, otherwise the third root would move into the right-half s-plane and the system becomes unstable.

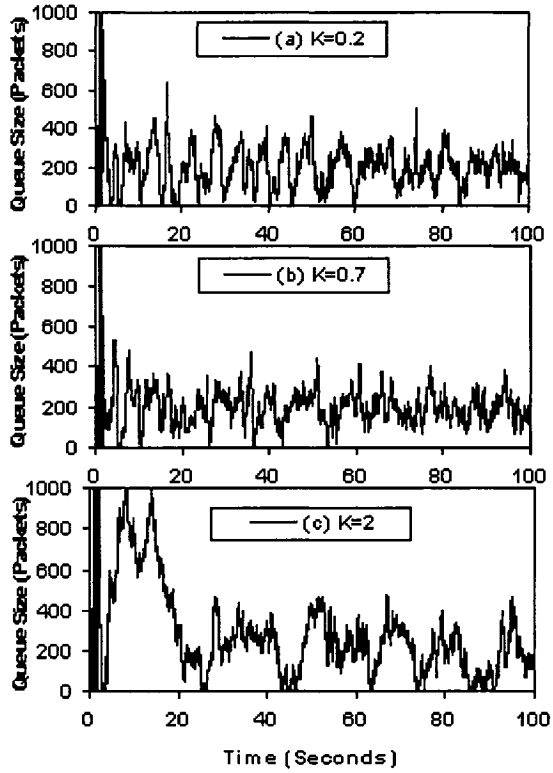

Fig. 3.19: Instantaneous queue size under the PI_PP controller with different damping ratios K (OM=ω_n =1, REF_QS=200, K=ξ=0.2, 0.7, 2)

Fig. 3.18 and Fig. 3.19 give the average queue size and the instantaneous queue size respectively. Comparing Fig. 3.19b (K=ξ=0.7) against Fig. 3.19a (K=ξ=0.2), one sees that, before the third root s_3 becomes dominant, the queue size oscillations decrease as ξ increases. This is also reaffirmed in the average queue size comparison between Fig. 3.18b and Fig. 3.18a (the overshoot of Fig. 3.18b is less than that of Fig. 3.18a), and expected from the control theoretical analysis [Ogat97]. However, when the third root becomes dominant (K=ξ=2), the oscillations (Fig. 3.19c) become larger and the settling time (Fig. 3.18c) becomes longer. From the control theoretical point of view, when the third root becomes

dominant and the system becomes less stable as the root moves closer to the origin of the *s*-plane, it becomes more difficult for the controller to clamp the queue size to the reference value of 200 packets. This can be seen in Fig. 3.18c that, even at the time of 100 seconds, the average queue size is still far from the reference value.

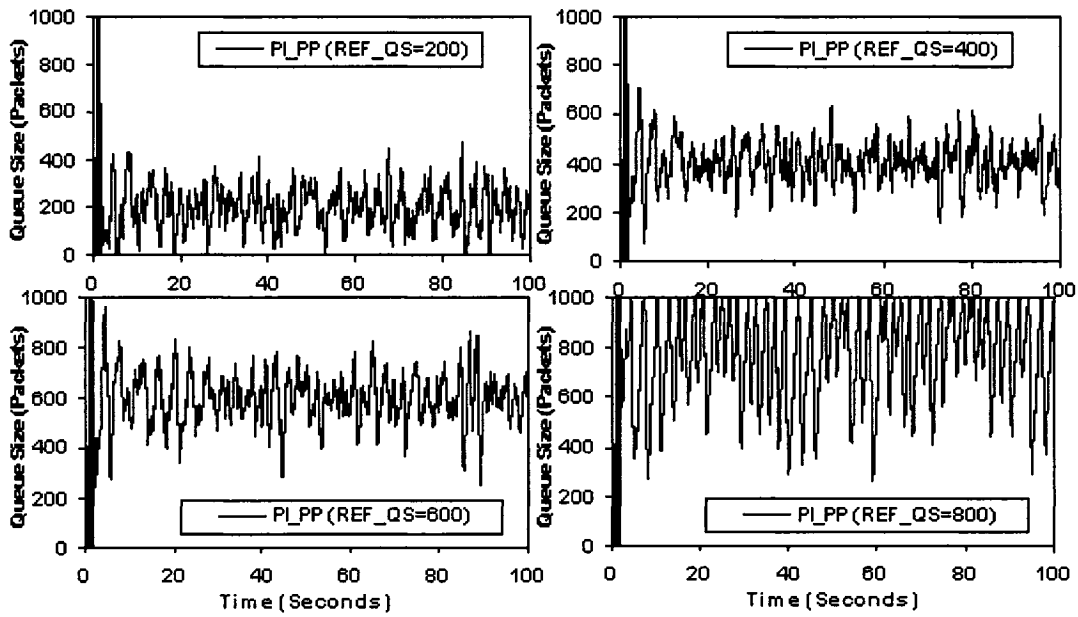

Fig. 3.20: Instantaneous queue size vs. Time: comparison of the PI_PP controller with different reference queue sizes (REF_QS=200, 400, 600, 800, OM=ω_n =1.8, K=ξ=0.5)

Experiment 4 - Different REF_QS

In this experiment, we choose different values of reference queue size (REF_QS).

Fig. 3.20 depicts the instantaneous queue size when REF_QS=200, 400, 600, 800 packets, while ξ and ω_n are set to 0.5 and 1.8 respectively. It is shown that the PI_PP controller can successfully clamp the queue size at the reference value when REF_QS=200, 400, 600 packets. When the reference queue size becomes 800 packets, due to buffer size limitation, the instantaneous queue size shows large oscillations. It is well known that larger REF_QS means longer average packet delay but higher buffer utilization. To seek a tradeoff between delay and utilization, the recommended REF_QS is 20%~60% of the buffer size.

Experiment 5 - Time-varying RTT

In order to investigate the PI_PP performance under time-varying RTTs, we vary RTPDs as follows: 20ms (0~100sec), 480ms (100~200sec) and 250ms (200~300sec).

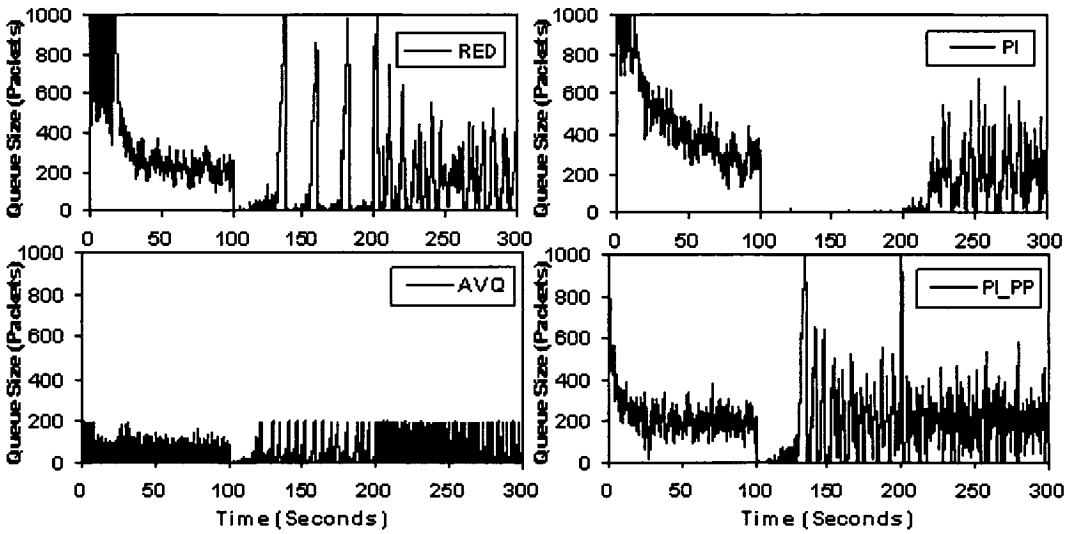

Fig. 3.21: Instantaneous queue size vs. Time: comparison of the RED, the PI, the AVQ and the PI_PP controllers (ξ=0.5, ω_n =1.8, REF_QS=200) in the case of time-varying RTT

The instantaneous queue size is given in Fig. 3.21. In the period 0~100 sec, all the four controllers can stabilize their queue size within finite time: 30 seconds for the RED, 70 seconds for the PI controller and 12 seconds for both the AVQ and the PI_PP controller. In the period 100~200 sec, the RED has difficulty to clamp its queue size to any steady-state value, and the PI controller also has difficulty to settle the average queue size on the pre-specified value of 200 packets. On the other hand, the PI_PP controller can achieve this easily. While the AVQ can clamp the queue size to a low average value, it is far from the REF_QS (200 packets). When it comes to the period 200~300 sec, all four controllers can stabilize the queue size again, but within different times: 7 seconds for the RED, 16 seconds for the PI and almost zero for the AVQ and the PI_PP. However, the AVQ does not have the capability to clamp the queue size around the pre-specified reference value of 200 packets.

Experiment 6 - Different RTT for different connections

Now let different connections have different RTTs. The RTPDs are set as: 100ms (ftp conn. 1~25), 200ms (ftp conn. 26~50), 300ms (ftp conn. 50~75), 400ms (ftp conn. 75~100), 100ms (http conn. 1~10) and 400ms (http conn. 11~20).

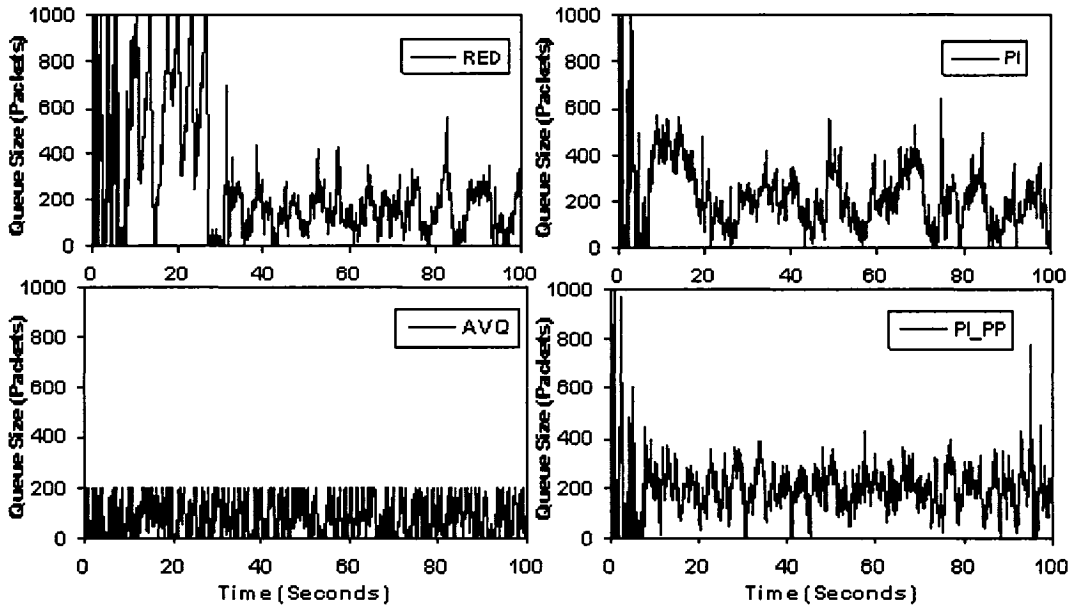

Fig. 3.22: Instantaneous queue size vs. Time: comparison of the RED, the PI, the AVQ and the PI_PP controllers in the case of different RTTs for different connections

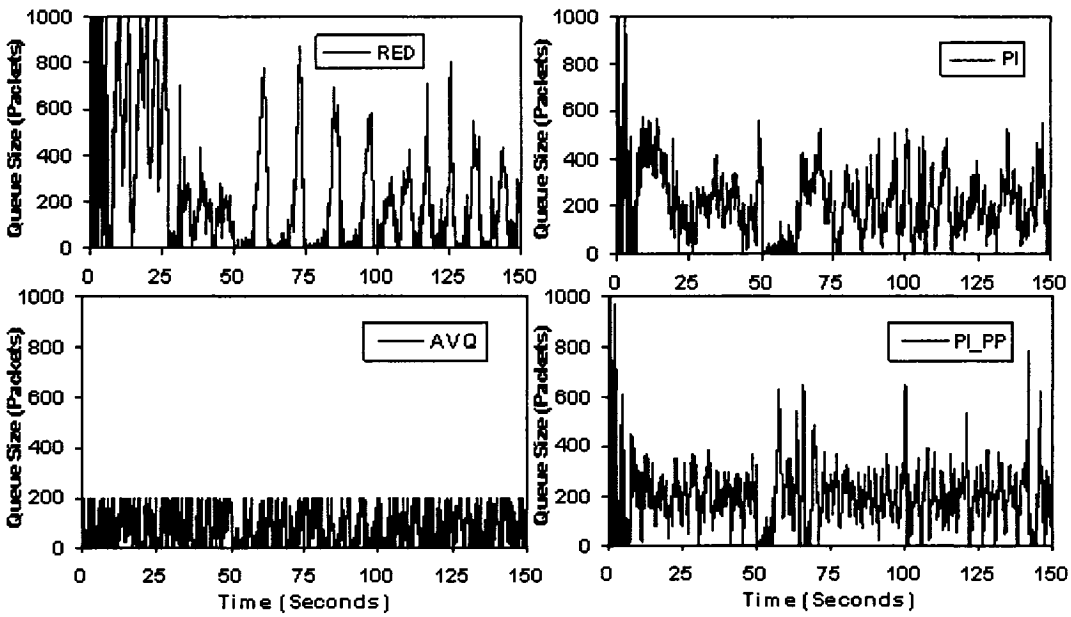

Fig. 3.23: Instantaneous queue size vs. Time: comparison of the RED, the PI, the AVQ and the PI_PP controllers (ξ=0.5, ω_n =1.8, REF_QS=200) in the case of different time-varying RTTs for different connections

The simulation result is given in Fig. 3.22. All four algorithms can stabilize the queue size, but it takes different times: 30 seconds for the RED, 7 seconds for the PI controller and the PI_PP controller, and almost zero for the AVQ. However the AVQ does not have the capability to clamp the average queue size to the reference value of 200 packets. The queue

size oscillations under the PI controller are larger than those under the PI_PP controller and the AVQ.

Experiment 7 - Different time-varying RTT for different connections

Now different time-varying RTTs for different connections are used in the experiment. The RTPDs configuration is the same as in Section 3.2 Experiment 6.

The instantaneous queue sizes for the four controllers are shown in Fig. 3.23. In the first simulation period (0~50 sec), it takes the RED 30 seconds to stabilize the queue size, whereas the PI controller and the PI_PP controller take 6 seconds, the AVQ takes almost zero second to drive the queue size to the steady state. In the second period (50~100sec), it becomes very difficult for the RED to clamp the queue size, while the PI controller, the AVQ and the PI_PP controller can stabilize the queue size quite well, but with different settling times: 12 seconds for the PI, 7 seconds for the AVQ and 4 seconds for the PI_PP. In the last simulation period (100~150 sec), the RED still has difficulty in stabilizing the queue size, whereas the PI controller and the PI_PP controllers can easily clamp the queue size to the average value of 200 packets which is the pre-specified reference queue size, and the AVQ can drive the queue size around a steady-state value (100 packets), which is far from the REF_QS of 200 packets.

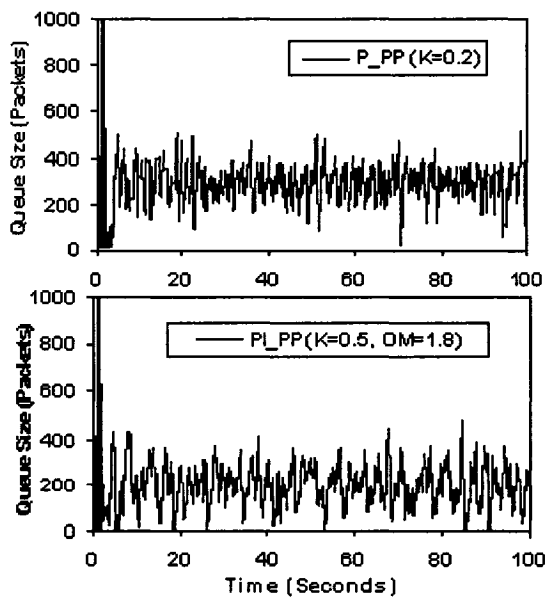

Fig. 3.24: Instantaneous queue size vs. Time:
Comparison of the P_PP controller and the PI_PP controller

Experiment 8 - Comparison of the PI_PP controller with the P_PP controller

To compare the PI_PP controller with the P_PP controller, we reproduce Fig. 3.24 from the previous diagrams. Although their settling times are comparable from Fig. 3.24, one obvious advantage of the PI_PP controller over the P_PP controller is that the former can clamp the average queue size to a preset value (e.g. 200 packets), while the latter shows large offset between the REF_QS (150 packets in our simulation) and the steady-state average queue size. Also in the PI_PP controller, ξ and ω_n are decoupled, namely they can be set separately as desired; while in the P_PP controller, ξ and ω_n are coupled by (3.13). In addition there is no severe limitation like condition (3.15) on the range of ξ in the PI_PP controller as is in the P_PP controller. Therefore we could conclude that all the limitations of the P_PP controller mentioned at the beginning of Section 3.3 are addressed in the PI_PP controller.

3.5 Concluding Remarks

This chapter proposes the P_PP controller and the PI_PP controller design for AQM routers based on the pole placement technique of the classical control theory. The formal P_PP controller and the PI_PP controller design procedures are presented and justified. The damping ratio ξ and/or undamped natural frequency ω_n are appropriately chosen such that: a) the transient response of the system is satisfied; b) all the poles lie in the left-half s-plane such that the stability of the system is guaranteed. Under the PI_PP controller, the average queue size can be clamped to the reference value. The effectiveness of the proposed controllers is verified via OPNET simulations by comparing with other algorithms.

The assumption (3.2) and the equation (3.29) are playing an important role in designing the pole-placement controllers. The assumption (3.2) simplifies the plant, and makes it possible to achieve the closed-form pole placement controllers. Choosing the third pole in the form of (3.29) has provided advantages in: a) the value m in (3.29) can now affect the system stability, e.g. $m>0$ and $\omega_n>0$ ensure the system stability; b) choosing $m>1$ can ensure that the pair of complex conjugate poles become the dominate poles; c) the form of (3.29) makes it easier to solve the PI_PP controller in a closed-form because the value m can now be determined in a closed-form as specified by (3.33).

Chapter 4 Self-Tuning Controllers

The controllers proposed in Ch. 3 have the disadvantages that, when significant load changes occur in the network, the system will show sluggish response or even become unstable if the controller parameters are not on-line tunable. Therefore, in this chapter, we shall exploit the results already established in Ch. 3 to propose two self-tuning controllers: the ST_P_PP controller and the ST_PI_PP controller. We incorporate the self-tuning feature to pole placement controllers and thus formulate the design procedure for the ST_P_PP controller and the ST_PI_PP controller. The performance evaluations for those two controllers are also presented in this chapter.

4.1 Design of the ST_P_PP Controller

The idea of the ST_P_PP controller is as follows. Initially we determine the controller parameter using the procedure presented in the P_PP controller design. Then we keep on-line monitor of the value of ξ. By "on-line monitor", we mean upon each packet arrival, we check the value of ξ. If ξ deviates from a preset interval (ξ_1, ξ_2) due to variations of network conditions, then we set ξ back to its initial value and use it to tune controller parameter again using the procedure given in Section 3.1.3.

To make the algorithm more general, we let the preset interval (ξ_1, ξ_2) be adjustable by the user. But here we still give some suggestions on the acceptable values of the interval. When setting the interval, note that larger ξ means longer settling time and milder overshoot [Ogat97]. Lots of simulation results (e.g. the results in Section 3.2, Experiment 2) show that as long as we keep $\xi \in (0.2, 0.8)$, the system shows satisfactory performance. The initial value for ξ can be chosen as the middle values of the interval, i.e. 0.5.

Next we need to derive the formula for the on-line monitor of ξ. From (3.7) and (3.8), we have

$$\xi = \frac{a}{2\omega_n} \quad \text{and} \quad \omega_n^2 = b \ ,$$

which gives us

$$\xi = \frac{\frac{1}{R_0} + \frac{2N}{R_0^2 C} - \frac{C^2 R_0}{2N} K_P}{2\sqrt{\frac{2N}{R_0^3 C} + \frac{C^2}{2N} K_P}} \quad . \tag{4.1}$$

Equ. (4.1) is the expression we need to monitor ξ.

The ST_P_PP controller design algorithm can now be presented as follows:

1) Initial setting:

1a) Choose and store the initial value of ξ. The recommended initial value is ξ=0.5.

1b) Use the procedure given in Section 3.1.3, calculate K_P.

2) On-line tuning:

2a) Choose an acceptable preset interval for ξ, the recommended range is $\xi \in (0.2, 0.8)$.

2b) Use (4.1) to monitor ξ, i.e. upon each packet arrival, calculate ξ using (4.1).

2c) Whenever ξ deviates from the preset interval due to variations of network conditions, set ξ back to its initial value, recalculate K_P using the procedure presented in Section 3.1.3.

Another parameter we need to specify for the ST_P_PP controller is the reference queue size (REF_QS). Similar to Section 3.1.3, the recommended REF_QS is 10%~50% of the buffer size. Please note that the meaning of *self-tuning* is that this procedure can *recalculate (tune) the controller parameter K_P* when the damping ratio ξ needs to be set to its initial value.

4.2 Performance Evaluation for the ST_P_PP Controller

The ST_P_PP controller is also verified via simulations using OPNET Modeler. The simulation setup is similar to the default setting as described in Section 3.2 except the following difference. The simulation length is set at 200 seconds. From 0 to 100 seconds, we use 100 greedy ftp sources and 20 http sources. While from 100 seconds to 200 seconds, we have 700 active sources (600 ftp sources and 100 http sources). The REF_QS is set at 150 packets.

Table 4.1 ST_P_PP controller parameter tuning

Time (seconds)	N	ξ	K_P
$0\sim100^-$	100	0.5	1.334×10^{-5}
100	600	1.06	1.334×10^{-5}
$100^+\sim200$	600	0.5	1.015×10^{-4}

Table 4.1 gives the design parameter for the ST_P_PP controller. Initially we set ξ=0.5, using the procedure presented in Section 3.1.3, and the controller parameters K_P is calculated to be 1.334×10^{-5}. When N changes from 100 to 600 at the time of 100 seconds, if K_P remains unchanged, the on-line monitor reports that the value of ξ becomes 1.06, which falls out of the following preset interval: $\xi \in (0.2, 0.8)$. Therefore the ST_P_PP controller tunes parameters K_P again to set ξ back to 0.5, which is its initial value, and K_P is tuned at 1.015×10^{-4}.

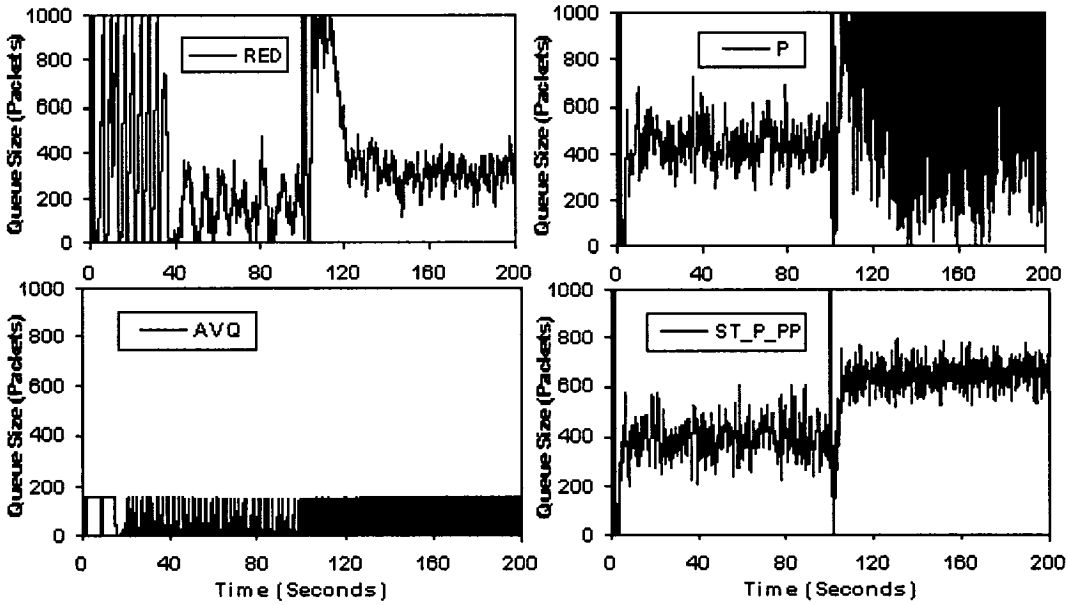

Fig. 4.1: Instantaneous queue size vs. Time: comparison of the RED, the P, the AVQ and the ST_P_PP controllers

Fig. 4.1 gives the instantaneous queue size. The figure shows that before N changes (i.e., from 0 second to 100 seconds), the RED has the longest settling time, whereas the P controller, the AVQ and the P_PP controller shows similar settling time. In addition, the steady-state queue size of the P controller is slightly larger than that of the ST_P_PP controller, whereas the AVQ shows the lowest steady-state queue size. When the load N changes from 100 to 600 at 100 seconds, the RED can still stabilize the queue size, but with the settling time of approximately 20 seconds; whereas the P controller cannot stabilize the queue size anymore, which shows very large oscillations. On the contrary, the ST_P_PP controller can promptly stabilize the queue size to a new value, with a settling time of 5 seconds which is shorter than that of the RED. The AVQ can drive the queue size to a steady-sate value, but at the cost of large packet loss rate, which is shown in the packet loss

rate comparison diagram Fig. 4.2.

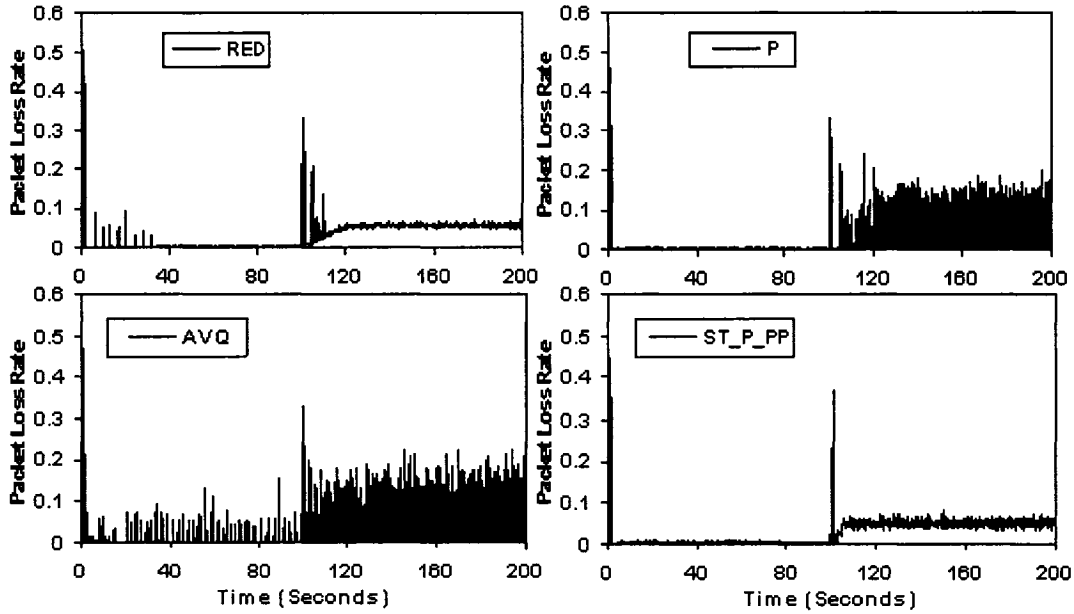

Fig. 4.2: Packet loss rate vs. Time: comparison of the RED, the P, the AVQ and the ST_P_PP controllers

Fig. 4.2 shows that, after the initial start-up period and until the end of the first simulation period (0^+~100 sec), the ranges of packet loss rates for the RED, the P controller, the AVQ and the ST_P_PP controller are 0%~10%, 0%~1%, 0%~15% and 0%~1%, respectively. When the load changes from 100 to 600 at t=100 seconds, all four controllers show a peak loss rate value around 35%. When the packet loss rates become steady-state in the second simulation period from 100 to 200 sec, the variations in packet loss rate for the RED and the ST_P_PP controller are similar: with the range from 5% to 7%, but the AVQ and the P controllers show a significant larger range from 0% to 25%.

4.3 Design of the ST_PI_PP Controller

Since the ST_P_PP controller is a proportional controller, therefore all the limitations of the P_PP controller described at the beginning of Section 3.3 still exist for the ST_P_PP controller. Namely the range of ξ is limited by condition (3.15); the values of ξ and ω_n are coupled as depicted by (3.13); and an offset exists between the REF_QS and the actual steady-state average queue size. To address these issues, we study the ST_PI_PP controller in this section.

Similar to the ST_P_PP controller, the idea of the ST_PI_PP controller is as follows. Initially we determine the controller parameters using the procedure presented in the PI_PP controller design. Then we monitor the values of ξ and ω_n on-line. If ξ or ω_n deviates from the preset intervals of (ξ_1, ξ_2) or (ω_{n1}, ω_{n2}) due to variations in network conditions, we set ξ and ω_n back to their initial values and use them to tune controller parameters again using the procedure given in Section 3.3.3.

In the next three subsections, we first derive the formula for on-line monitor of ξ and ω_n. Then we present the process to determine the acceptable preset intervals for ξ and ω_n. Finally the ST_PI_PP controller design algorithm is given.

4.3.1 Deriving the Formulae for On-line Monitor of ξ and ω_n

From (3.24), (3.25) and (3.26), we have

$$\xi\omega_n = \frac{a}{2+m} \quad , \tag{4.2}$$

$$\omega_n^2 = b - 2m\left(\frac{a}{2+m}\right)^2 \quad , \tag{4.3}$$

$$\omega_n^2 = \frac{c(2+m)}{am} \quad , \tag{4.4}$$

where a, b, c and m are defined in Section 3.3.1. By equating (4.3) to (4.4), the following equation can be obtained,

$$(ab-c)m^3 + (4ab-2a^3-6c)m^2 + (4ab-12c)m - 8c = 0 \quad . \tag{4.5}$$

The smallest positive real root of (4.5) is the valid value of m to calculate ξ and ω_n.

From (4.4) and (4.2), we can derive

$$\omega_n = \sqrt{\frac{c(2+m)}{ma}} \quad , \tag{4.6}$$

$$\xi = \sqrt{\frac{ma^3}{c(2+m)^3}} \quad . \tag{4.7}$$

Thus (4.5), (4.6) and (4.7) are used to monitor ξ and ω_n.

Before we keep on-line monitor of ξ and ω_n, we need to pre-assign acceptable intervals for ξ and ω_n, such as (ξ_1, ξ_2) and (ω_{n1}, ω_{n2}). To make the algorithm more general, we let the preset intervals (ξ_1, ξ_2) and (ω_{n1}, ω_{n2}) be adjustable by the user. But we shall still give some

suggestions on the acceptable values for the intervals as presented in the next subsection.

4.3.2 Determining Acceptable Intervals for ξ and ω_n

The control theory [Ogat97] determines that ξ and ω_n must be within some appropriate ranges in order to have a good tradeoff among the system response speed, the maximum overshoot and the stability margin. Keeping this in mind, we determine the acceptable intervals of ξ and ω_n through simulations. We do this by fixing one parameter (say ξ) while varying the other. The simulation configuration is the same as that presented in Section 2.3.1. For the controller parameters design, we use the procedure presented in Section 3.3.3. Lots of simulations are performed, but here we only show some representative results for brevity.

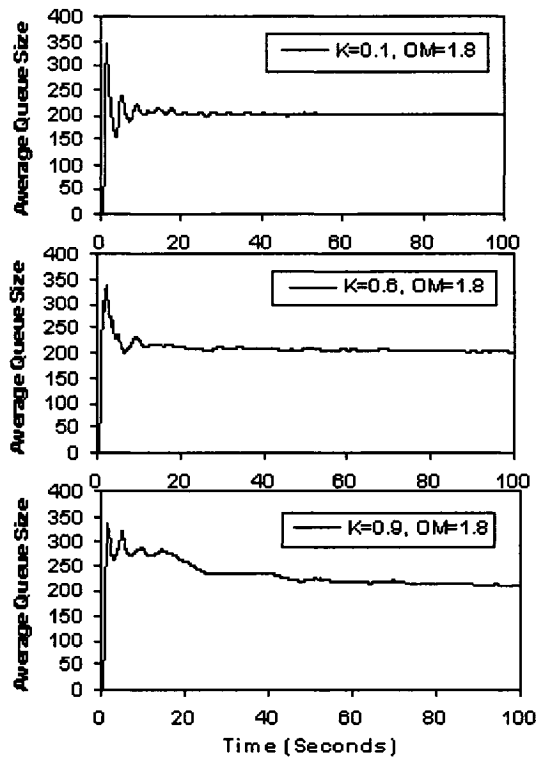

Fig. 4.3: Average queue size vs. Time: Determining the acceptable interval of the damping ratio K (ξ) (OM=ω_n =1.8, REF_QS=200, K=ξ=0.1, 0.6, 0.9)

To determine the acceptable interval of ξ, we fix ω_n at the value of 1.8, and adjust ξ to different values of 0.1, 0.6 and 0.9. Fig. 4.3 shows the average queue size evolution. It is shown that when ξ increases, the maximum overshoot decreases and the response time increases. Namely, when ξ is small (e.g. K=ξ=0.1), the maximum overshoot becomes large; while if ξ is large (e.g. K=ξ=0.9), the response time also gets large. To seek a tradeoff, we

determine that the acceptable interval for ξ is (0.2, 0.8), and the initial value of ξ is taken as the middle value, i.e. 0.5, of the interval.

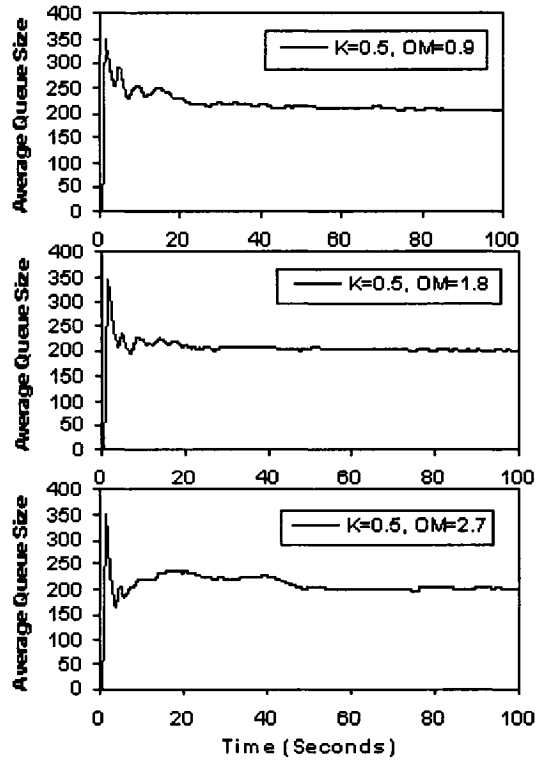

Fig. 4.4: Average queue size vs. Time: Determining the acceptable interval of the undamped natural frequency OM (ω_n) (K=ξ=0.5, REF_QS=200, OM=ω_n =0.9, 1.8, 2.7)

Likewise, to determine the acceptable interval for ω_n, we fixed ξ at the value of 0.5 and change ω_n to different values of 0.9, 1.8 and 2.7. Fig. 4.4 gives the average queue size evolution. The simulation result shows that when ω_n is too small (e.g. 0.9) or too large (e.g. 2.7), the response time gets larger[10] with regard to the response time when ω_n is 1.8. Therefore if we require response time to be within a reasonable range (e.g. from 0 to 40 seconds), we determine that the acceptable interval for ω_n is (1.0, 2.6), and the initial value of ω_n is set as the middle value, i.e. 1.8, of the interval.

Therefore we estimate that as long as we keep $\xi \in (0.2, 0.8)$ and $\omega_n \in (1.0, 2.6)$, the system shows satisfactory performance. The initial values for ξ and ω_n are chosen as the middle values of the intervals, i.e. 0.5 and 1.8 for ξ and ω_n respectively.

[10] In Section 3.4 Experiment 2, it is concluded that the settling time decreases as ω_n increases for a fixed value of ξ. But here we find that when ω_n is too large (e.g. ω_n=2.7), the settling time increases. That is because when ω_n=2.7, we have m=0.95<1, which makes the third root become the dominant pole.

4.3.3 The ST_PI_PP Controller Design Algorithm

Now we can present the ST_PI_PP controller design algorithm as follows:

1) Initial setting:

1a) Choose and store the initial value of ξ and ω_n, e.g. set $\xi = \xi_0$ and $\omega_n = \omega_{n0}$.

1b) Use the procedure given in Section 3.3.3, calculate K_P and K_I.

2) On-line tuning:

2a) Choose acceptable preset intervals for ξ and ω_n, e.g. set $\xi \in (\xi_1, \xi_2)$ and $\omega_n \in (\omega_{n1}, \omega_{n2})$.

2b) Upon each packet arrival, use (4.5), (4.6) and (4.7) to monitor ξ and ω_n.

2c) Whenever ξ or ω_n deviates from the preset intervals due to variations of network conditions, set ξ and ω_n back to their initial values, recalculate K_P and K_I using (3.30), (3.32) and (3.33). The recalculated values of K_P and K_I are adopted as the new controller parameters.

As justified by section 4.3.2, we have the following recommended values: ξ_0=0.5, ω_{n0}=1.8, ξ_1=0.2, ξ_2=0.8, ω_{n1}=1.0, ω_{n2}=2.6. Similar to the ST_P_PP controller, another parameter we need to specify for the ST_PI_PP controller is the REF_QS. Since we have the integral operation in the ST_PI_PP controller, the offset between the REF_QS and the steady-state average queue size can be removed eventually. The recommended REF_QS is 20%~60% of the buffer size.

4.4 Performance Evaluation for the ST_PI_PP Controller

The ST_PI_PP controller is verified via OPNET simulations. The simulation setup is similar to that in Section 3.4, except that, the simulation length, and the settings for N, R_0 may be different, and these differences will be described in particular experiments. For the ST_PI_PP controller we set REF_QS to 200 packets by default, other values of REF_QS may be used when necessary.

Experiment 1 - Comparing RED, PI, AVQ and ST_PI_PP

First we compare the RED, the PI controller, the AVQ and the ST_PI_PP controller when varying N. The simulation length is set as 200 seconds. From 0 to 100 seconds, we use 100 greedy ftp sources and 20 http sources. While from 100 seconds to 200 seconds, we have 700 active sources (600 ftp sources and 100 http sources).

Table 4.2 gives the design parameters for the ST_PI_PP controller. From Table 4.2, initially ξ=0.5, ω_n =1.8, using the procedure in Section 3.3.3, the controller parameters K_P and K_I are calculated as 1.095×10^{-5} and 8.526×10^{-6} respectively; when N changes from 100 to

600 at the time 100 seconds, if K_P and K_I remain unchanged, the on-line monitor reports that the values of ξ and ω_n become 1.00 and 2.85 respectively, all deviate from the following preset intervals: $\xi \in (0.2, 0.8)$ and $\omega_n \in (1.0, 2.6)$. Therefore the ST_PI_PP controller tunes parameters K_P and K_I again to set ξ and ω_n back to 0.5 and 1.8 respectively, which are their initial values.

Table 4.2 The ST_PI_PP controller parameter tuning when varying N

Time (seconds)	N	ξ	ω_n	K_P	K_I
$0\sim100^-$	100	0.5	1.8	1.095×10^{-5}	8.526×10^{-6}
100	600	1.00	2.85	1.095×10^{-5}	8.526×10^{-6}
$100^+\sim200$	600	0.5	1.8	4.802×10^{-5}	1.334×10^{-4}

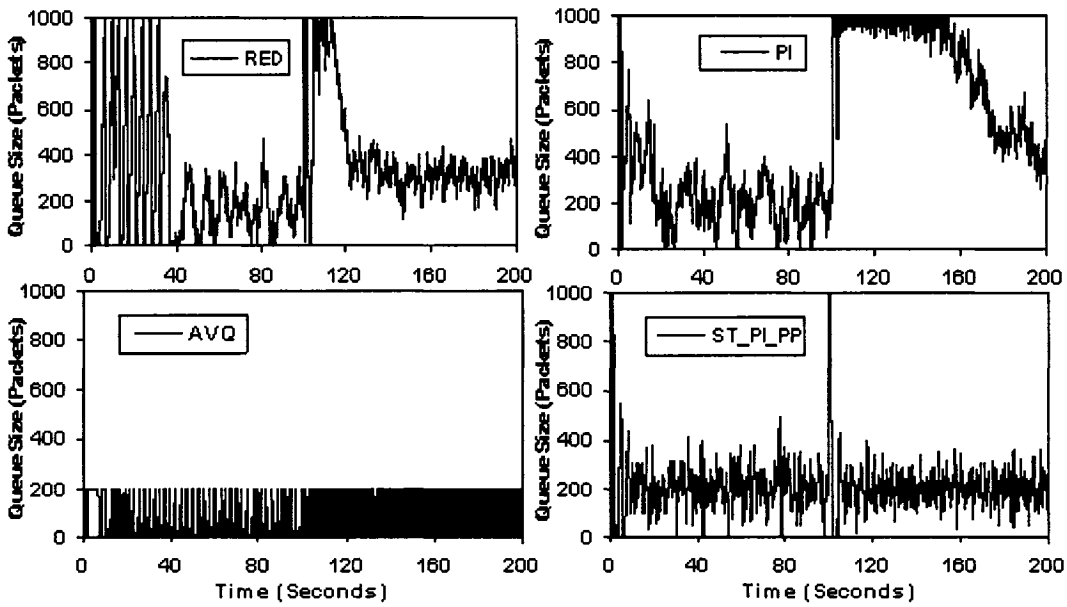

Fig. 4.5: Instantaneous queue size vs. Time: comparison of the RED, the PI, the AVQ and the ST_PI_PP controllers when varying N (REF_QS=200)

Fig. 4.5 gives the time evolution of the instantaneous queue size. It is shown that when the load N changes from 100 to 600 at 100 seconds, the PI controller shows extremely long settling time (longer than 100 seconds), with the RED the second longer (approximately 40 seconds), while the ST_PI_PP controller shows that its settling time is close to zero. The AVQ also shows a zero settling time, but at the cost of large packet loss rate, which can be inferred from Fig. 4.6. Moreover, one can conclude that after the load changes, the average queue size under the RED increases, and the PI controller or the AVQ has difficulty to clamp the queue size to the specified reference value of 200, while the ST_PI_PP can easily clamp

the queue size to the reference value. Therefore the ST_PI_PP controller gives the best system performance among the four controllers when the load changes.

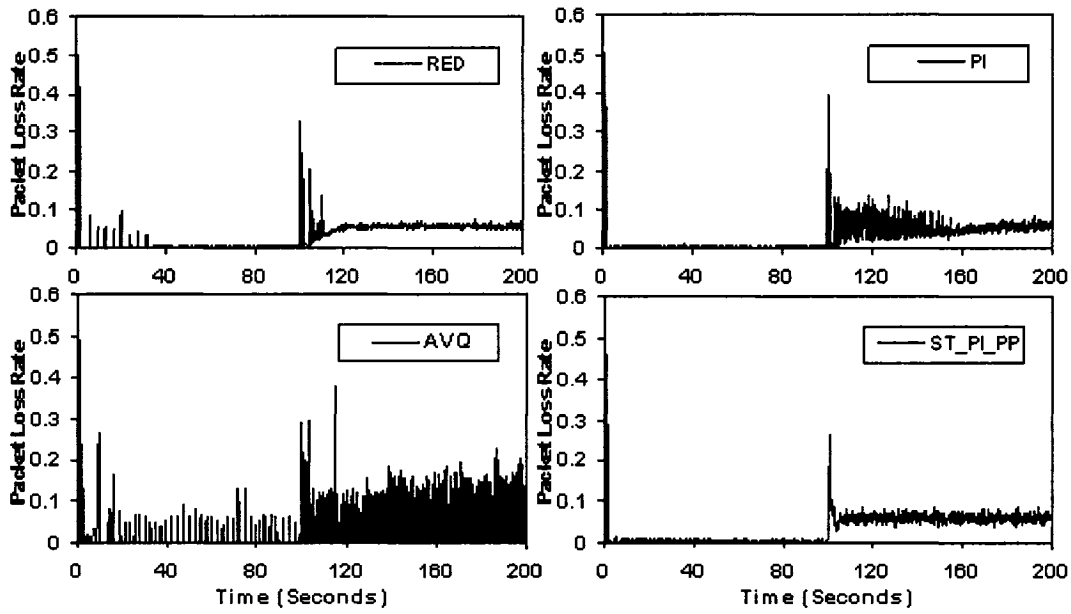

Fig. 4.6: Packet loss rate vs. Time: comparison of the RED, the PI, the AVQ and the ST_PI_PP controllers

Fig. 4.6 presents the packet loss rate under the four controllers. At the beginning of the simulation ($t=0^+$), the RED, the PI controller and the AVQ show a peak value of 50%, whereas the ST_PI_PP controller presents a relatively smaller one (45%). From 0 to 100 seconds, the PI controller and the ST_PI_PP controller give similar loss rate variations, within the range from 0 to 1%. On the other hand, the RED controller shows a relatively larger range from 0 to 10%, and the AVQ shows the largest range from 0 to 25%. After the load changes from 100 to 600 at t=100 seconds, we could obtain the following observations from Fig. 4.6: a) The RED, the PI controller, the AVQ and the ST_PI_PP controller exhibit peak values of 33.2%, 39.5%, 38% and 26.9% respectively; b) the durations needed for the packet loss rates to achieve steady-state values are different for the four controllers: 20 seconds for the RED, 60 seconds for the PI controller, 15 seconds for the AVQ, 10 seconds for the ST_PI_PP controller; c) when the packet loss rates become steady-state, the variations in packet loss rate for the RED, the PI controller and the ST_PI_PP controller are similar: with the range from 5% to 7%, but the AVQ shows a significant larger range from 0% to 23%.

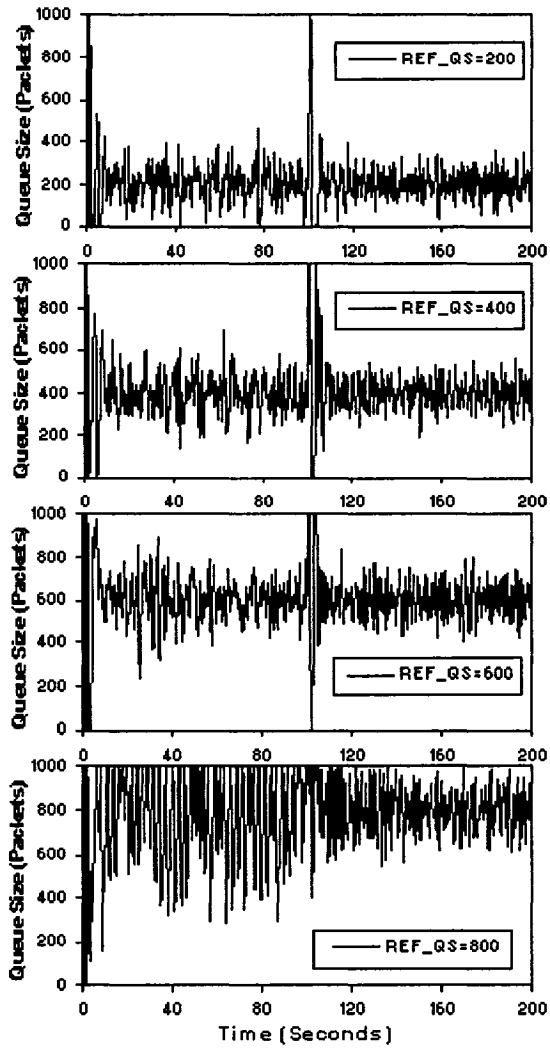

Fig. 4.7: Instantaneous queue size vs. Time: comparison of the ST_PI_PP controller with different reference queue sizes (REF_QS=200, 400, 600, 800)

Experiment 2 - Different REF_QS

In this experiment, we choose different values of reference queue size (REF_QS). The load change is the same as in the previous experiment, i.e. at the time of 100 seconds, the number of ftp connections (N) changes from 100 to 600 and the number of http connections also varies from 20 to 100.

Fig. 4.7 depicts the instantaneous queue size when REF_QS = 200, 400, 600, 800 packets, while ξ and ω_n are set to 0.5 and 1.8 respectively. It is shown that even under significant load changes, the ST_PI_PP controller can still successfully clamp the queue size at the reference value when REF_QS=200, 400, 600 packets. When the reference queue size

becomes 800 packets, due to buffer size limitation, the instantaneous queue size shows large oscillations. It is well known that larger REF_QS means longer average packet delay but higher buffer utilization. To seek a tradeoff between delay and utilization, the recommended REF_QS is 20%~60% of the buffer size.

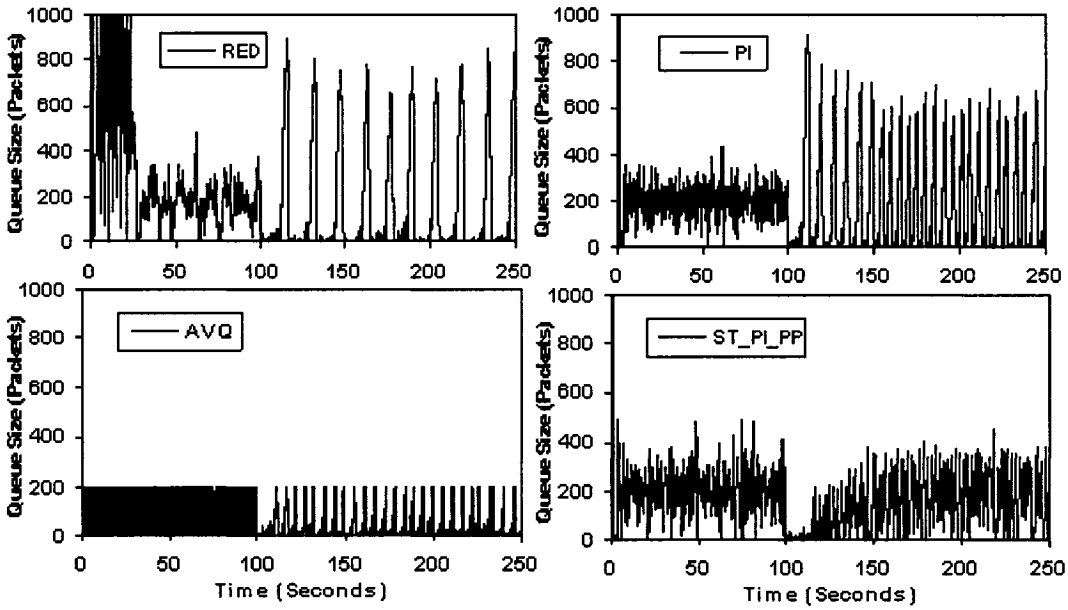

Fig. 4.8: Instantaneous queue size vs. Time: comparison of the RED, the PI, the AVQ and the ST_PI_PP controllers when varying RTPD (REF_QS=200)

Table 4.3 The ST_PI_PP controller parameter tuning when varying the RTPD

Time (Seconds)	RTPD (ms)	ξ	ω_n	K_P	K_I
0~100$^-$	120	0.5	1.8	1.470×10^{-5}	4.904×10^{-5}
100	360	0.49	2.72	1.470×10^{-5}	4.904×10^{-5}
100$^+$~250	360	0.5	1.8	6.332×10^{-6}	4.299×10^{-7}

Experiment 3 - Time-varying RTT

Now we vary the RTPDs (round trip propagation delays) of all connections. N is kept constant at 100. But RTPDs are changed from 120 ms to 360 ms at the simulation time of 100 seconds. The simulation length is set as 250 seconds.

Table 4.3 gives the design parameters for the ST_PI_PP controller. Note that RTPDs are varied at the time of 100 seconds. The on-line monitor is reporting the value of 2.72 for ω_n, which has fallen out of the preset interval of (1.0, 2.6).

Fig. 4.8 depicts the instantaneous queue size under the four controllers. It is shown that

after the variation of RTPDs, the RED and the PI controller have difficulties in stabilizing the queue size, while the ST_PI_PP controller and the AVQ can achieve this after a certain amount of time: 45 seconds for the ST_PI_PP controller, 15 seconds for the AVQ. However, the AVQ does not have the capability to clamp the queue size to the reference value of 200 packets.

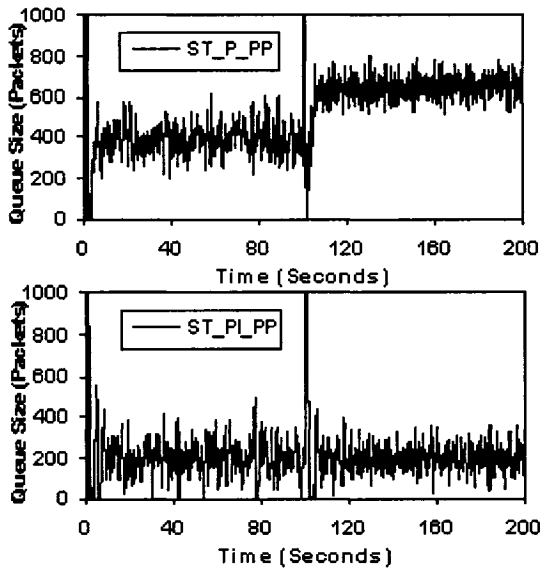

Fig. 4.9: Instantaneous queue size vs. Time: comparison of the ST_P_PP controller and the ST_PI_PP controller

Experiment 4 - Comparison of the ST_PI_PP controller with the ST_P_PP controller

To make a comparison of the ST_PI_PP controller with the ST_P_PP controller, we reproduce Fig. 4.9 from the previous diagrams. Although their settling times are comparable from Fig. 4.9, one obvious advantage of the ST_PI_PP controller over the ST_P_PP controller is that the former can clamp the average queue size to a preset value (e.g. 200 packets), while the latter shows large offset between the REF_QS (150 packets in our simulation) and the steady-state average queue size. Also in the ST_PI_PP controller, ξ and ω_n are decoupled, namely they can be set separately as desired; while in the ST_P_PP controller, ξ and ω_n are coupled by (3.13). In addition there is no severe limitation like condition (3.15) on the range of ξ in the ST_PI_PP controller as is in the ST_P_PP controller. Therefore we could conclude that all the limitations of the ST_P_PP controller mentioned at the beginning of Section 4.3 are addressed in the ST_PI_PP controller.

4.5 Concluding Remarks

In this chapter, we propose the ST_P_PP controller and the ST_PI_PP controller design for AQM routers. The formal design procedures for these two controllers are justified and presented. The transient response and stability of the system are guaranteed by appropriately choosing damping ratio ξ and/or undamped natural frequency ω_n. The ST_P_PP controllers have the capability to keep on-line tuning of the controller parameter K_p when ξ deviates from the preset interval, and thus to ensure the stability and satisfactory system performance even under significant load changes. The ST_PI_PP controller can assign proper intervals of both ξ and ω_n to achieve good AQM performance and thereby make the system adapt to significant load changes very well. In addition the ST_PI_PP controllers can clamp the steady-state queue length to a specified reference value. Furthermore, under the ST_PI_PP controller, ξ can be chosen to be independent of ω_n, and vice versa. This makes it more flexible to adjust the oscillations, the overshoots and the settling time. We verify the effectiveness of the controllers via simulations using OPNET.

Chapter 5 AQM Design Based on $\mathcal{H}_\infty$ Optimal Control

We continue our efforts of finding good AQM controllers by exploiting different control theoretical approaches. Unlike the classical controllers, using the pole locations, we use weight functions in the $\mathcal{H}_\infty$ optimal control problem to determine the AQM controller. We first present the control method of $\mathcal{H}_\infty$ *S/U* Mixed Sensitivity Problem (MSP). We then propose the design of the *S/U* MSP controller and the *S/T/U* MSP controller. Finally the investigation on the selection of weight functions and the performance evaluations are provided.

Appendix B gives some detailed preliminaries on the $\mathcal{H}_\infty$ optimal control theory, including the introduction of the three forms (*S/U*, *S/T* and *S/T/U*) of the MSP, the general control configuration, and the state-space solution for the $\mathcal{H}_\infty$-optimization problem.

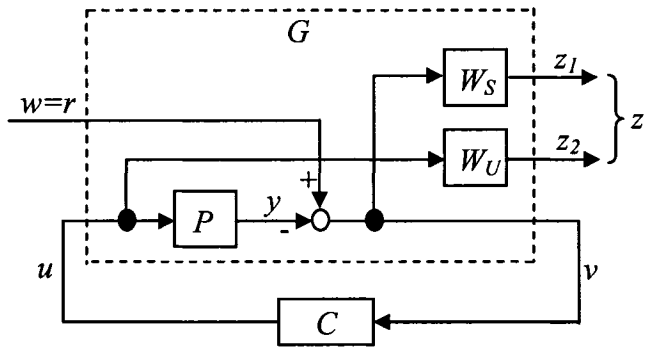

Fig. 5.1: *S/U* MSP in general control configuration

5.1 Control Method of the $\mathcal{H}_\infty$ *S/U* MSP

Fig. 5.1 depicts the configuration of our *S/U* MSP controller. Here, the exogenous input w is the reference queue size in the TCP/AQM system in Fig. 2.2; the exogenous output z_1 is the weighted queue size error (difference between the reference queue size and the actual queue size); the exogenous output z_2 is the weighted drop probability variation; the controller input v is the queue size error; the control signal u is the drop probability variation; the generalized controller C is the *S/U* MSP controller we are trying to synthesize; and G is the generalized plant model, which can be determined by P (the plant of the TCP/AQM

system, i.e. $P_1(s)e^{-sR_0}$ in Fig. 2.2), the sensitivity weight W_S, and the input sensitivity weight W_U.

As summarized in Appendix B and [SkPo01], one of our performance requirements is that the $\mathcal{H}_\infty$-norm of the weighted sensitivity $W_S S$ must be less than one. Physically, it means that the shape of magnitude curve of the sensitivity function S is upper bounded by $1/|W_S|$. Similarly, to achieve robustness or to restrict the magnitude of the plant input signals, we may place an upper bound $1/|W_U|$ on the magnitude of the input sensitivity function U, and W_U is named as the input sensitivity weight function. To combine these "mixed sensitivity" specifications, we have to formulate the following S/U Mixed Sensitivity Problem (MSP): to solve for an optimal controller C_{opt} such that

$$C_{\text{opt}} = \underset{C \text{ stabilizes } P}{\text{minimize}} \left\| \begin{bmatrix} W_S S \\ W_U U \end{bmatrix} \right\|_\infty . \tag{5.1}$$

Fig. 5.1 determines G from W_S, W_U, and the plant model P. Therefore after we choose the functions W_S, W_U, and with P available, the $\mathcal{H}_\infty$ S/U MSP (5.1) can then be solved with the μ-toolbox in MATLAB, which implements the state-space solution [DoGl89].

5.2 Design of the S/U MSP Controller

From Section 2.2, the plant transfer function of the TCP/AQM control system is given by

$$P(s) = \frac{\frac{C^2}{2N} e^{-sR_0}}{(s + \frac{2N}{R_0^2 C})(s + \frac{1}{R_0})} . \tag{5.2}$$

In this plant, there is a time delay term e^{-sR_0}. A system with a time delay (or dead time) is an infinite-dimensional system and not representable as a rational transfer function. For a state-space realization it must therefore be approximated. An nth-order approximation of a time delay J may be obtained by putting n first-order Padé approximations [Wong92] in series

$$e^{-Js} \approx \frac{(1 - \frac{J}{2n} s)^n}{(1 + \frac{J}{2n} s)^n} .$$

For the TCP/AQM control system, the plant input is δp which is the variation of the

probability of packet drop around the operating point. We know that drop probability must be bounded in the range [0, 1]. That means we must put some bounds on the plant input. Therefore we will solve the *S/U* MSP or the *S/T/U* MSP to design our AQM controller. In the *S/T* MSP, since we cannot bound the plant input u, it is not considered in this thesis.

The general control configuration for the *S/U* mixed sensitivity optimization is given in Fig. 5.1. For the TCP/AQM control system, the plant is described by (5.2). To get the generalized plant in Fig. 5.1, we need to specify the weight functions W_S and W_U. Actually a difficulty that sometimes arises with $\mathcal{H}_\infty$ control is the selection of weights such that the $\mathcal{H}_\infty$ optimal controller provides a good trade-off between conflicting objectives in various frequency ranges. In the *S/U* MSP controller design for the TCP/AQM system, we have the following insights on the selection of weights W_S and W_U:

1) The input sensitivity function U is the transfer function from the reference input r to the plant input u. Therefore, if we want to limit the plant input, we could accordingly choose the value of W_U, since the magnitude of U is upper bounded by $1/|W_U|$. Specifically, in the AQM controller design, W_U is used to bound the plant input δp which is the variation of packet drop probability around the operating point p_0, where p_0 [HoMi01a] is equal to $2N^2 / (R_0 \times C)^2$. The variation δp should be in the order of p_0 minus 2 or 3 if we desire a stable system. Therefore W_U can be chosen in the order of $1/p_0$ plus 2 or 3. For example, let x be the order of p_0, i.e. $p_0 \approx 10^x$, then δp should be in the order of $x - 2$ or $x - 3$, and W_U can be chosen in the order of $-x + 2$ or $-x + 3$. This criterion has also been verified through simulations which will be illustrated in the design example below;
2) The sensitivity function S is the transfer function from the reference input to the error between the plant output and the reference input. A common choice [SkPo01] for the weight W_S is

$$W_S = \frac{s/M + \omega_B}{s + \omega_B A} \quad , \tag{5.3}$$

where M is the upper bound for the maximum peak magnitude of S; ω_B is the minimum bandwidth frequency and A is the maximum steady-state tracking error. Selecting $A << 1$ ensures approximate integral action with $S(0) \approx 0$. A larger value of ω_B yields a faster response and a worse robustness; the recommendation ranges for the selection of M, ω_B and A will be presented at Section 5.3.5.

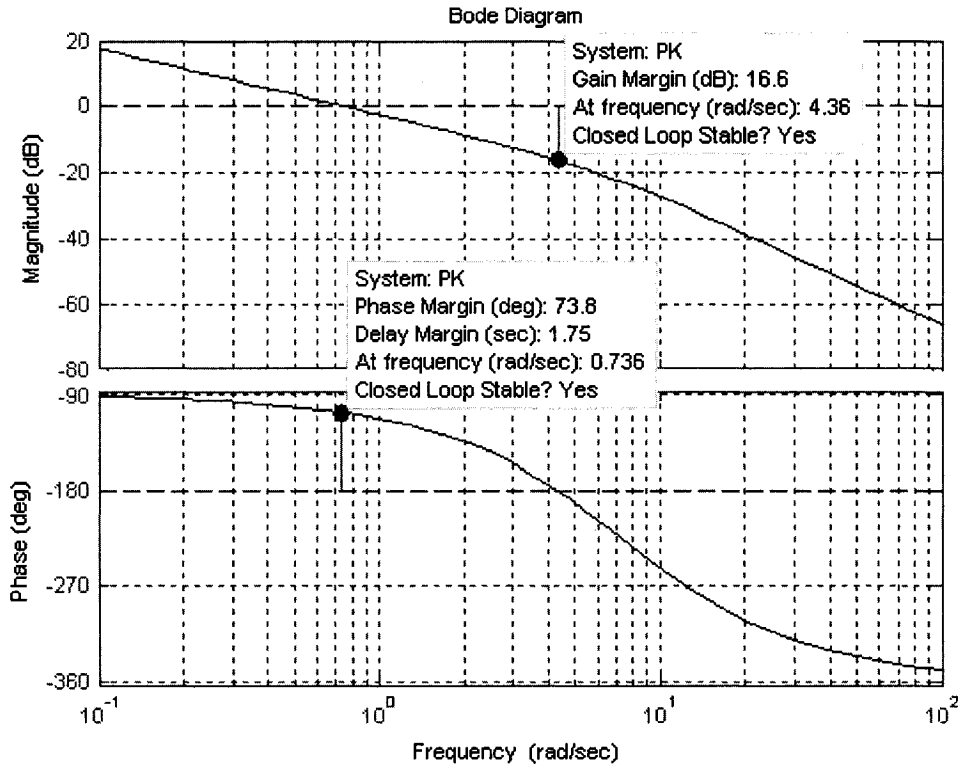

Fig. 5.2: Bode diagram of the open-loop transfer function PC (system PK) under the S/U MSP controller

Here we provide a design example on the selection of the weight functions and its impacts on the performance of the TCP/AQM system. We use the default network settings as presented in Section 3.2, i.e., N=100, R_0=0.25 second, and C=9708 packets/second. We design the AQM controller by solving the S/U MSP. W_S is chosen as in (5.3) and tentatively we choose $M = 1.5$, $\omega_B = 3.5$, $A = 10^{-4}$. As mentioned before, W_U can be chosen in the order of $1/p_0$ plus 2 or 3, where p_0 is equal to $2N^2 / (R_0 \times C)^2 = 3.395 \times 10^{-3}$. Namely $1/p_0$ is in the order of 3; therefore W_U can be chosen in the order of 5 or 6. Tentatively we choose $W_U = 5 \times 10^5$. Now that $P(s)$, $W_S(s)$, $W_U(s)$ have been determined, the $\mathcal{H}_\infty$ S/U mixed sensitivity problem can then be solved with the μ-toolbox in MATLAB, which gives us the following optimal $\mathcal{H}_\infty$ controller for the S/U MSP:

$$C(s) = \frac{(0.18936s^3 + 2.33475s^2 + 6.80856s + 1.99738) \times 10^{-6}}{2.05281 \times 10^{-7} s^4 + 0.0193389s^3 + 0.253058s^2 + 0.967258s + 3.38509 \times 10^{-4}} . \tag{5.4}$$

The optimal $\mathcal{H}_\infty$ norm value γ_{min} is equal to 4.9417.

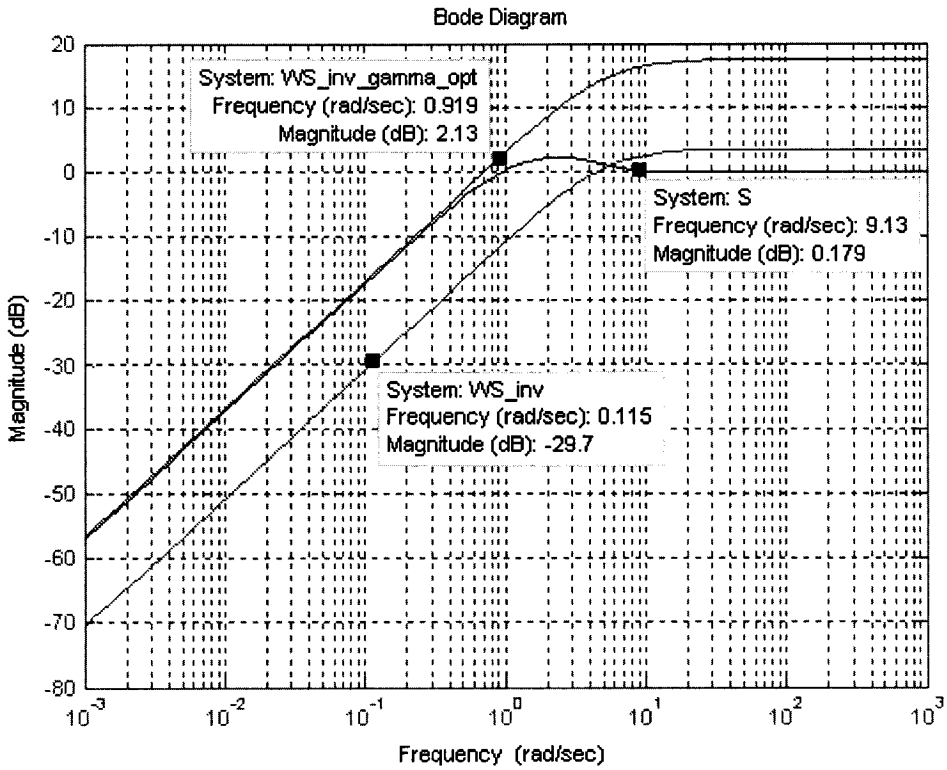

Fig. 5.3: Bode diagrams of S, $1/W_S$ (system WS_inv) and γ_{min}/W_S (system WS_inv_gamma_opt)

Fig. 5.2 gives the Bode diagram of the open loop transfer function PC. From the figure, we conclude that the Phase Margin is 73.8° at frequency 0.736 rad/sec, and the Gain Margin is 16.6 dB at frequency 4.36 rad/sec. Therefore the system is closed-loop stable. Fig. 5.3 presents the Bode diagrams of the sensitivity function S, the inverse of the sensitivity weight function $1/W_S$ and γ_{min}/W_S. It is shown that S is small at low frequencies, which means the sensitivity of the system output to disturbance is small and the error between system output and the reference input is small at low frequencies. The figure also shows that the magnitude of S reaches the asymptotic value 1 (0dB) at high frequencies. So far, we have mentioned that, $|S|$ is upper bounded by $1/|W_S|$. But in a strict sense, $|S|$ should be upper bounded [SkPo01] by $\gamma_{min}/|W_S|$ (the proof is provided in Appendix E), and this is verified by Fig. 5.3.

Similarly the Bode diagrams of the input sensitivity function U, the inverse of the sensitivity weight function $1/W_U$ and γ_{min}/W_U are illustrated in Fig. 5.4. It is shown that U is in the order of –5 (-100dB), and $|U|$ is upper bounded by $\gamma_{min}/|W_U|$.

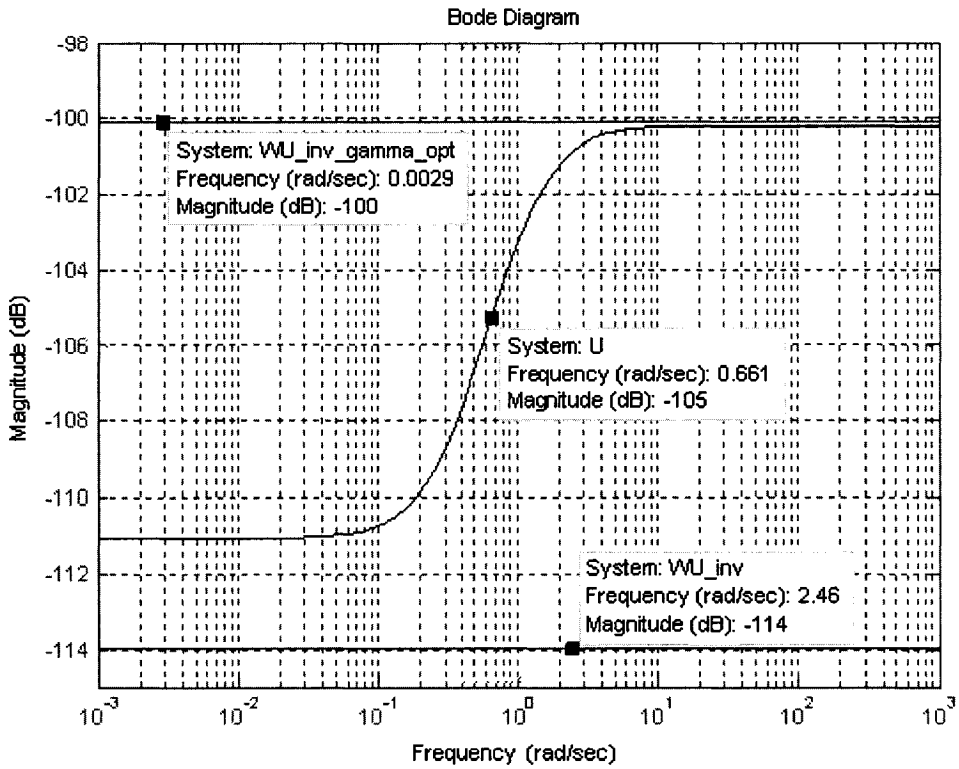

Fig. 5.4: Bode diagram of U, $1/W_U$ (system WU_inv) and γ_{min}/W_U (system WU_inv_gamma_opt)

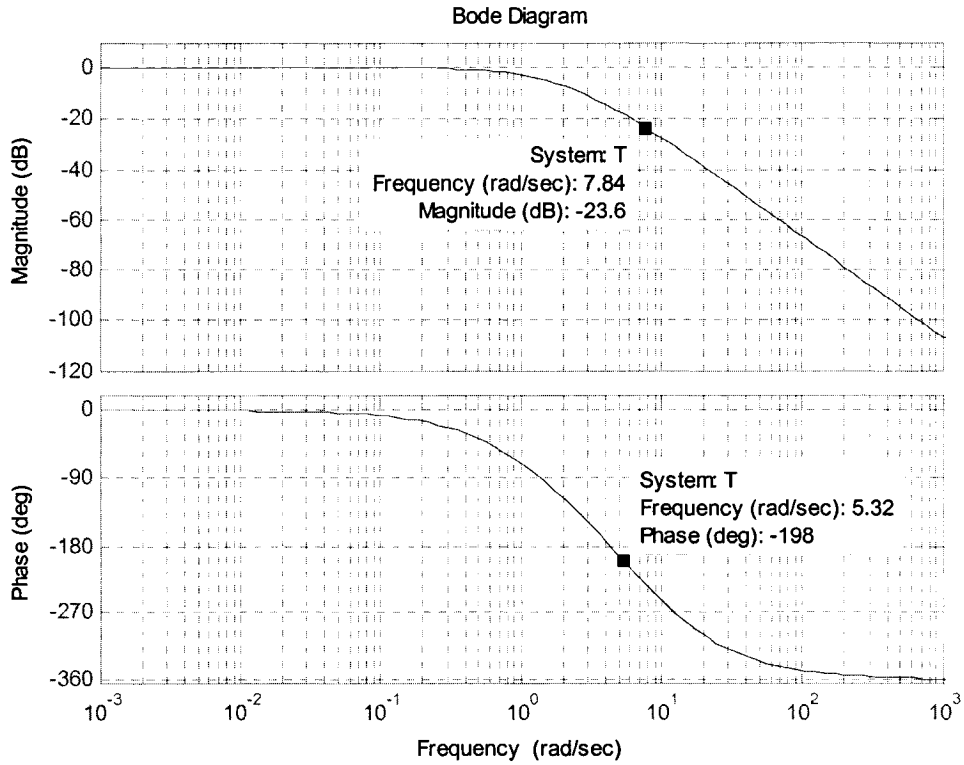

Fig. 5.5: Bode diagram of T under the S/U MSP controller

Fig. 5.5 presents the Bode diagram of the complementary sensitivity function T. It is shown that the magnitude of T starts to roll off at the frequency of 1 rad/sec. We shall refer to this diagram in Section 5.6 later on when we compare the S/U MSP with the $S/T/U$ MSP.

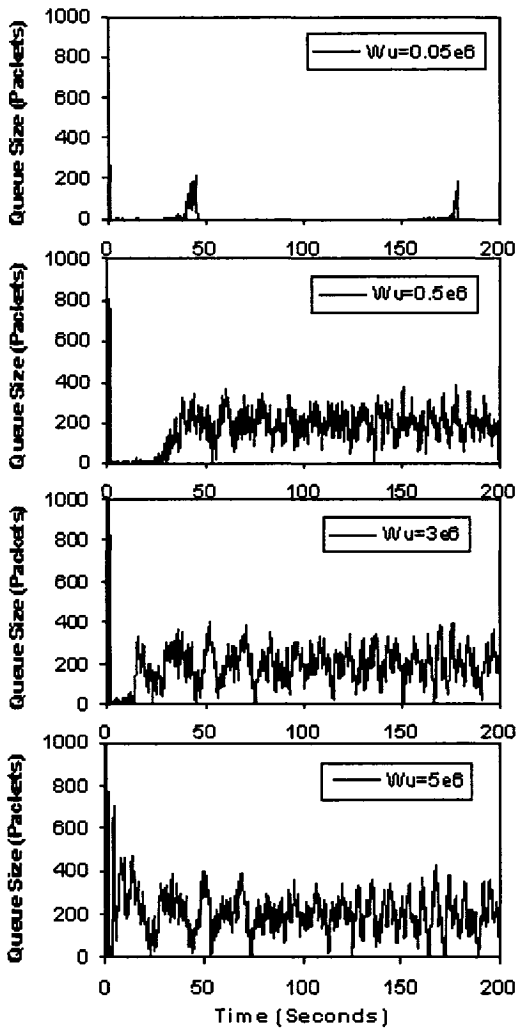

Fig. 5.6: Instantaneous queue size vs. Time: comparison of different W_U

5.3 Selection of Weight Functions

The optimal controller (5.4) is implemented in OPNET simulations. The simulation setup and results are given in Section 5.5. Moreover, to make some recommendations on the selection ranges of the parameters of the weight functions, we have performed extensive simulations to investigate the impacts of the parameter settings of the weight functions. Specifically, we choose different values of W_U, ω_B, M, A and explore their impacts on the system performance. We do this by varying one parameter (say W_U) and fixing the other three parameters (say ω_B, M and A).

5.3.1 Varying W_U

We fix the value of M at 1.5, ω_B at 5.5, A at 10^{-4}, and set W_U to different values: 0.05×10^6, 0.5×10^6, 3×10^6, 5×10^6. By solving the S/U MSP with the μ-toolbox in MATLAB, different $\mathcal{H}_\infty$ optimal controllers are obtained corresponding to different values of W_U.

In our example, the reference queue size is set as 200 packets. Fig. 5.6 illustrates the instantaneous queue size under different $\mathcal{H}_\infty$ optimal controllers corresponding to different W_U. When W_U is too small (e.g. 0.05×10^6), since the plant input δp (variation of drop probability) is too large considering the fact that the input sensitivity is upper bounded by $\gamma_{min}/|W_U|$, the system has difficulty to get stable. The instantaneous queue size has difficulty to achieve a stable oscillation around the reference queue size of 200 packets when W_U $=0.05\times10^6$, as we can see from Fig. 5.6. When W_U is increasing, since the variation of drop probability δp is decreasing, the instantaneous queue size can converge to stable oscillations around the reference queue size faster. From Fig. 5.6 the settling time can be determined as 34 seconds, 14 seconds and 4 seconds when W_U is equal to 0.5×10^6, 3×10^6 and 5×10^6 respectively.

On the other hand, the γ_{min} values are 3.0448, 6.4227, 14.7926 and 20.0411 corresponding to $W_U=0.05\times10^6$, 0.5×10^6, 3×10^6 and 5×10^6, namely, the γ_{min} value gets larger when W_U increases. From [SkPo01], we know that γ_{min} relates to the stability robustness in the sense that stability robustness gets worse when γ_{min} increases. Therefore the stability robustness gets worse when W_U increases.

5.3.2 Varying ω_B

In this subsection, we investigate the impacts of ω_B by setting it to different values of 2.5, 3, 3.5 while fixing W_U, M and A at 0.5×10^6, 1.5 and 10^{-4} respectively. Different $\mathcal{H}_\infty$ optimal controllers are obtained corresponding to different values of ω_B.

Fig. 5.7 gives the instantaneous queue size evolutions under different ω_B. It is shown that settling time is equal to 101 seconds, 50 seconds and 6 seconds when ω_B is set to 2.5, 3 and 3.5 respectively. The γ_{min} is calculated as 4.1066, 4.5347 and 4.9417 when ω_B is equal to 2.5, 3 and 3.5 respectively, which means the stability robustness becomes worse when ω_B increases.

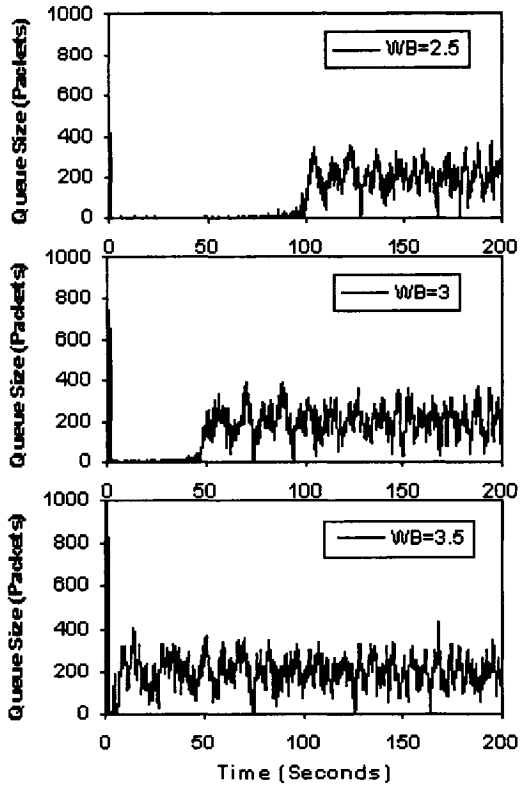

Fig. 5.7: Instantaneous queue size vs. Time: comparison of different ω_B

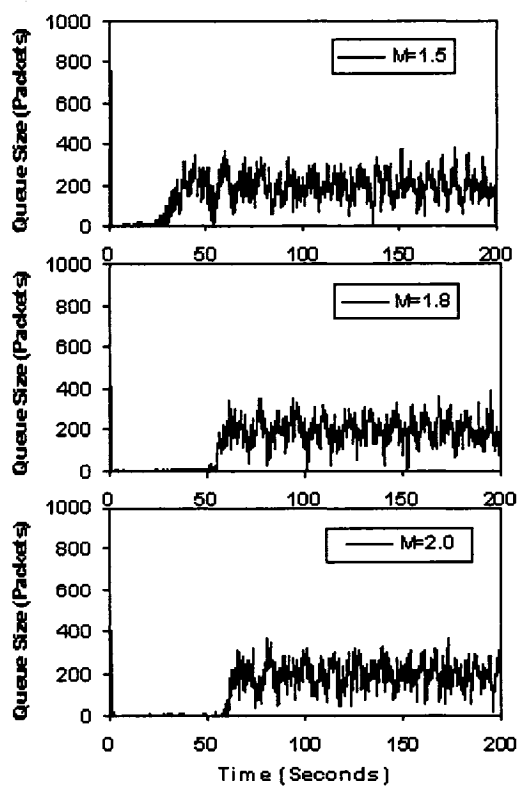

Fig. 5.8: Instantaneous queue size vs. Time: comparison of different M

5.3.3 Varying M

In order to explore the impact of M, we set M to different values of 1.5, 1.8 and 2.0 but fixing W_U, ω_B and A at 0.5×10^6, 5.5 and 10^{-4} respectively.

The instantaneous queue size evolutions under different M are illustrated in Fig. 5.8, from which the settling time can be determined as 34 seconds, 58 seconds and 62 seconds when M is set to 1.5, 1.8 and 2.0 respectively. The increasing of M does not have much effect on γ_{min} (6.4227, 6.4142 and 6.4105), which means no significant impacts on stability robustness.

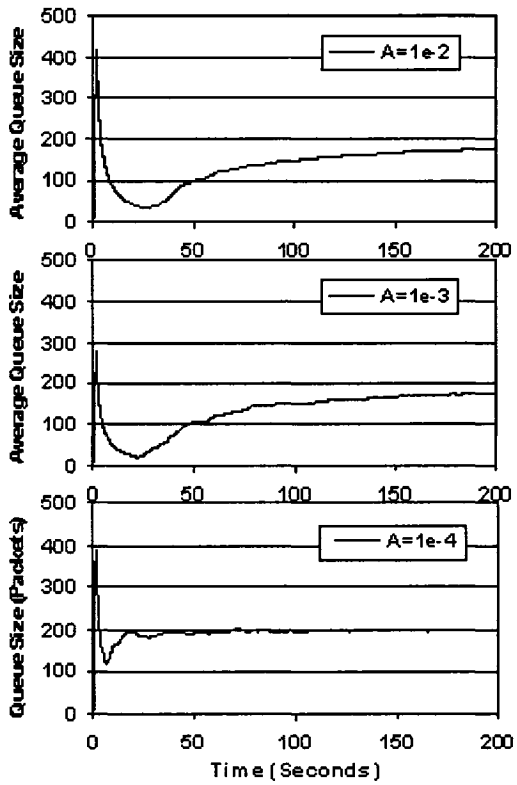

Fig. 5.9: Average queue size vs. Time: comparison of different A

5.3.4 Varying A

In this subsection, A is set to different values of 10^{-2}, 10^{-3} and 10^{-4}, while W_U, ω_B and M are kept unchanged at 0.5×10^6, 3.5 and 1.5 respectively.

The average queue size is given in Fig. 5.9. We define the steady-state error of the queue size to be the difference between the reference queue size and the steady-state average queue size. These errors can be determined from Fig. 5.9 to be 26.84 packets, 25.15 packets

and 2.15 packets when A is set to 10^{-2}, 10^{-3} and 10^{-4} respectively. This means the absolute value of the steady-state error decreases when A becomes lower. The decreasing of A does not impact the stability robustness much because γ_{min} values (4.8091, 4.9296 and 4.9417) show no significant differences.

5.3.5 Recommendation on the Selection of Weight Functions

From Section 5.3.1 –5.3.4, we could conclude the following guidelines regarding the choices of W_U, ω_B, M and A.

1) The order of W_U is equal to the order of $1/p_0$ plus 2 or 3, where p_0 is equal to $2N^2$ / $(R_0 \times C)^2$. Larger value of W_U induces worse robustness against uncertainties but faster response.
2) ω_B can be chosen within the range 2.5 ~ 5. Likewise, increasing ω_B causes worse robustness against uncertainties and faster response.
3) M is typically set as 1.5 ~ 2. Larger value of M yields slower response.
4) Usually we select $A \ll 1$, in the order of –4 or –5. Lower value of A ensures better tracking performance.

5.4 Design of the *S/T/U* MSP Controller

Recall that T is the transfer function from the negative value of noise $-n$ to plant output y. To reduce sensitivity to noise and uncertainty, we want T small at high frequencies, and so we may want additional roll-off in the open loop gain L. Therefore we may add $W_T T$ to the stack H, where W_T is the complementary sensitivity function, then the *S/U* MSP becomes the *S/T/U* MSP and will be discussed in this section. The general control configuration for the *S/T/U* mixed sensitivity optimization is given in Fig. B.5. The plant P is described by (5.2) for the TCP/AQM control system. To get the generalized plant in Fig. B.5, we need to specify the weight functions W_S, W_T and W_U. The selections of W_S and W_U have been discussed and evaluated in Section 5.2 and Section 5.3. In this section, we would investigate the selection of the weight function W_T.

By making $|W_T|$ small (e.g. smaller than 2.5) at low frequencies and large at high frequencies, the roll-off in the open loop gain L can be achieved. Typically we choose W_T as

$$W_T = \frac{m_1(1+m_2 s)}{1+m_3 s} \quad , \tag{5.5}$$

where $m_1 \leq 2.5$, $m_3 << m_2$. Here $m_1 \leq 2.5$ ensures that $|W_T|$ is small at low frequencies, while $m_3 << m_2$ guarantees that $|W_T|$ is large at high frequencies.

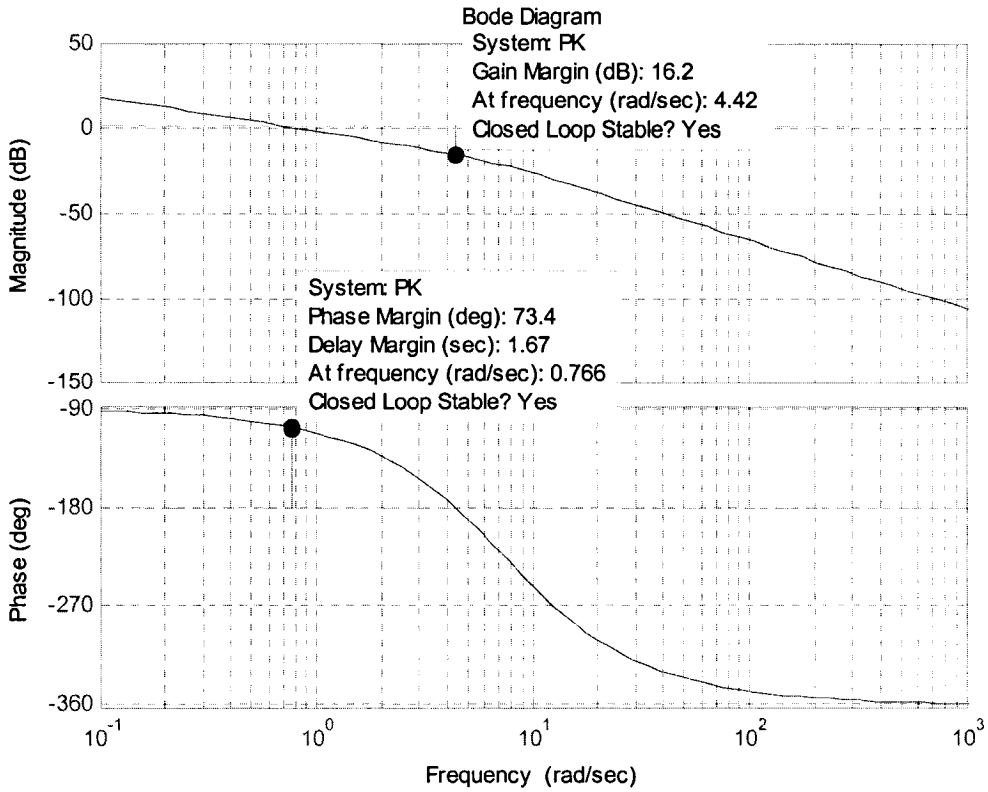

Fig. 5.10: Bode diagram of the open-loop transfer function PC (system PK) under the $S/T/U$ MSP controller

Next we discuss via an example the selection of the weight function W_T and its impact on the performance of the TCP/AQM system. The same network parameters and configuration are used here as in Section 3.2, i.e., N=100, R_0=0.25 second, and C=9708 packets/second. The AQM controller is designed by solving the $S/T/U$ MSP. W_S is chosen as in (5.3) and we choose $M = 1.5$, $\omega_B = 3.5$, $A = 10^{-4}$. As in the example in Section 5.2, W_U is set to 5×10^5. For W_T, (5.5) is used and we tentatively choose m_1=2.4, m_2=0.4 and m_3=10^{-3}. Now that P(s), W_S(s), W_U(s) and W_T(s)have been determined, the $\mathcal{H}_\infty$ $S/T/U$ mixed sensitivity problem can then be solved with the μ-toolbox in MATLAB, which gives the following optimal $\mathcal{H}_\infty$ controller for the above problem:

$$C(s) = \frac{(0.00019743s^4 + 0.19987s^3 + 2.4414s^2 + 7.1009s + 2.0825)\times 10^{-6}}{2.5313\times 10^{-10}s^5 + 1.8915\times 10^{-5}s^4 + 0.018909s^3 + 0.24785s^2 + 0.96861s + 3.3898\times 10^{-4}} \quad . \quad (5.6)$$

The optimal $\mathcal{H}_\infty$ norm value $\gamma_{\min}$ is equal to 5.3324.

Fig. 5.10 gives the Bode diagram of the open loop transfer function PC. From the figure, we conclude that the Phase Margin is 73.4° at frequency 0.766 rad/sec, and the Gain Margin is 16.2 dB at frequency 4.42 rad/sec. Therefore the system is closed-loop stable.

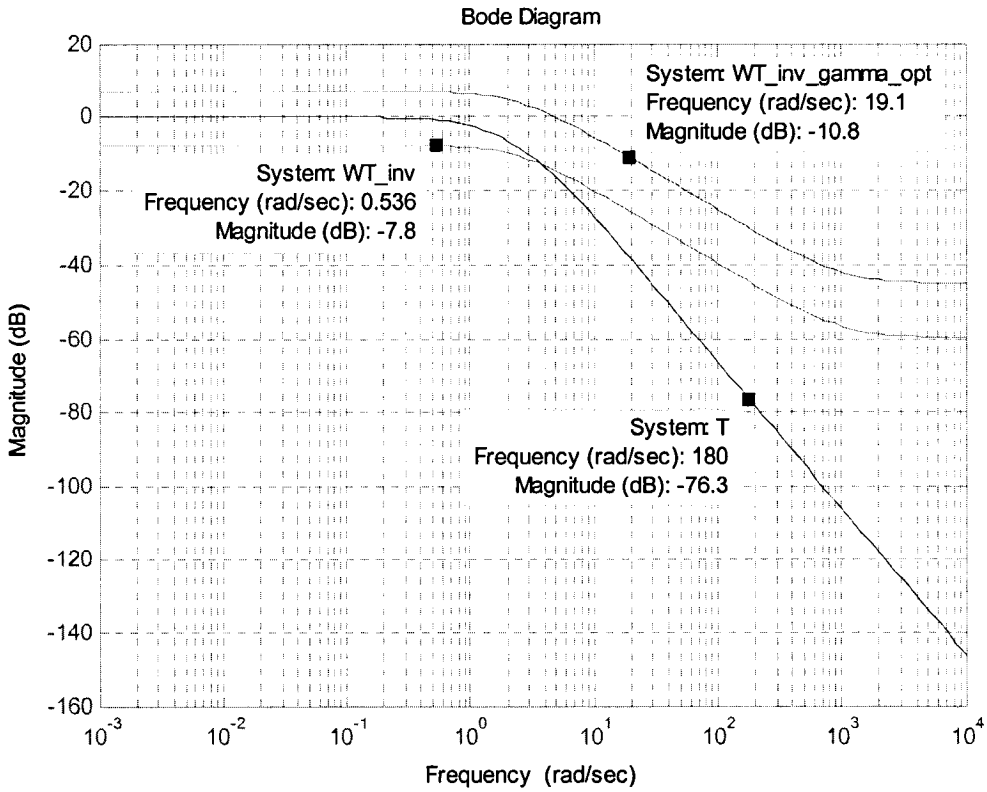

Fig. 5.11: Bode diagrams of T, $1/W_T$ (WT_inv) and γ_{min}/W_T (WT_inv_gamma_opt) under the $S/T/U$ MSP controller

Fig. 5.11 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 0dB at low frequencies, and drop off at high frequencies, which reduces the sensitivity to noise and uncertainty. So far, we have mentioned that, $|T|$ is upper bounded by $1/|W_T|$. Fig. 5.11 also verifies that $|T|$ is upper bounded by $\gamma_{min}/|W_T|$.

Similar to the work in Section 5.3, we have also performed extensive simulations to investigate the impacts of the parameter settings of W_T, with the intention to make some recommendations on the selection of the weight function W_T. Specifically, we choose different values of m_1, m_2 and m_3, and explore their impacts on the system performance. We do this by varying one parameter (say m_1) and fixing the other three parameters (say m_2 and m_3). From lots of simulations, the following guidelines are concluded regarding the choices

of m_1, m_2 and m_3 in the *S/T/U* MSP: m_1 and m_2 can be chosen within the range 0.5 ~ 2.5 and 0.1 ~ 1.0, respectively; m_3 is recommended to be set in the order of –3 or –4.

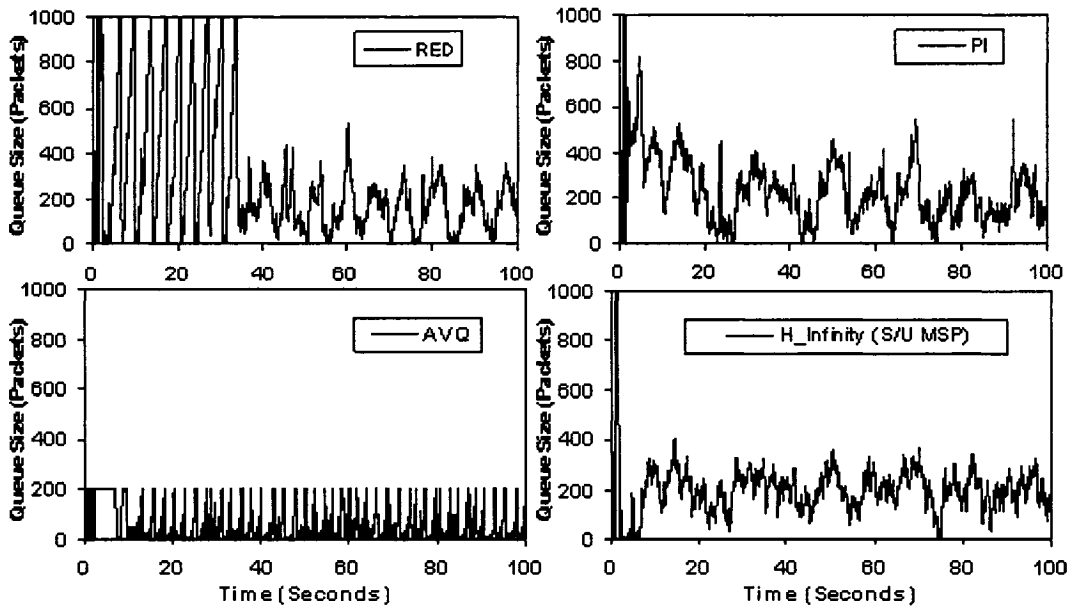

Fig. 5.12: Instantaneous queue size vs. Time: comparison of the RED, the PI, the AVQ and the $\mathcal{H}_\infty$ *S/U* MSP controllers

5.5 Performance Evaluation for the *S/U* MSP Controller

Both the *S/U* MSP controller (5.4) and the *S/T/U* MSP controller (5.6) have been verified via simulations using OPNET Modeler [OPNE01]. From the simulation results, the *S/U* MSP controller (5.4) and the *S/T/U* MSP controller (5.6) shows very similar performance. Therefore in this thesis we only present the simulation results for the *S/U* MSP controller. A comparison of these two controllers, including an explanation of the reason for the performance similarities and the necessity of the usage of the *S/T/U* MSP controller in some cases, is given in Section 5.6.

The typical bottleneck configuration described in Section 2.3.1 is used as our simulation model. Unless specified otherwise in each individual experiment, we use the default setting for the network parameters as described in Section 3.2. The REF_QS is set as 200 packets by default.

Experiment 1 – Comparison of RED, PI, AVQ and *S/U* MSP

This experiment compares the performance of the RED, the PI controller, the AVQ and the $\mathcal{H}_\infty$ *S/U* MSP controller (5.4). Fig. 5.12 gives the instantaneous queue size evolution comparison under these four controllers. It is shown that the settling time of the RED is

longer than that of the PI controller which in turn is longer than that of the AVQ and the *S/U* MSP controller. Fig. 5.12 also justifies the conclusion that the queue size oscillations of the RED and the PI controller are larger than that of the *S/U* MSP controller, which in turn is larger than that of the AVQ. In addition, both the PI controller and the *S/U* MSP controller can clamp the steady-state average queue size to the reference value of 200 packets, whereas the RED and the AVQ do not have this capability.

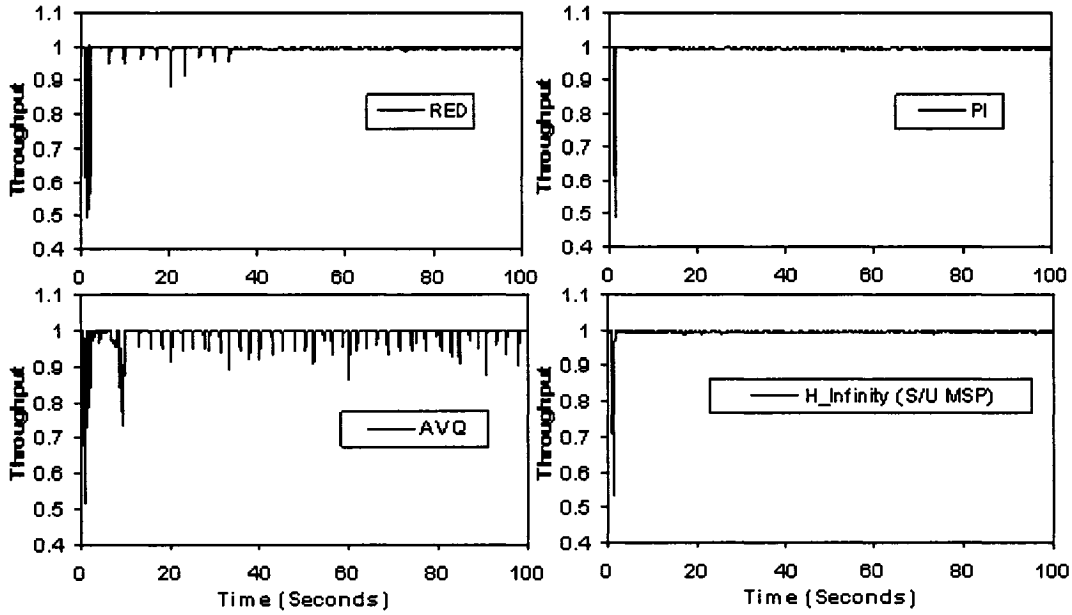

Fig. 5.13: Router throughput vs. Time: comparison of the RED, the PI, the AVQ and the $\mathcal{H}_\infty$ *S/U* MSP controllers

Fig. 5.13 presents the bottleneck router throughput under the four controllers. The router throughput is defined as the number of packets forwarded by the router divided by the number of packets entering the router within a specific period of time. At the beginning of the simulation, the RED, the PI controller and the AVQ show a bottom value of 0.5, whereas the *S/U* MSP controller presents a relatively higher one (0.54). After the initial start-up period, the PI controller and the *S/U* MSP give similar throughput variations, within the range from 0.995 to 1. On the other hand, the RED shows a relatively larger range from 0.89 to 1, and the AVQ shows the largest range from 0.74 to 1.

Experiment 2 – Different REF_QS

Now we investigate the case when the reference queue size (REF_QS) is set to different values under the *S/U* MSP controller.

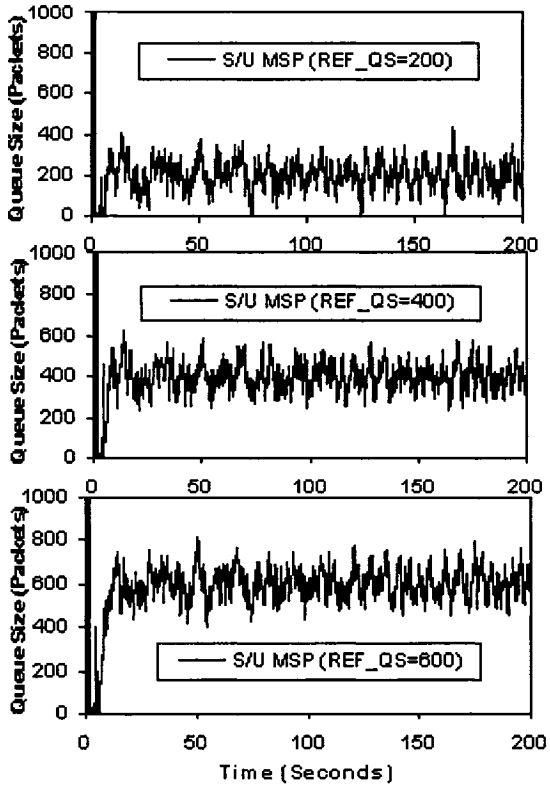

Fig. 5.14: Instantaneous queue size vs. Time: comparison of the *S/U* MSP controller with different reference queue sizes (REF_QS=200, 400, 600)

Fig. 5.14 depicts the instantaneous queue size when REF_QS=200, 400, 600 packets. It is shown that the *S/U* MSP controller can successfully clamp the queue size at the reference value. It is well known that larger REF_QS means longer average packet delay but higher buffer utilization. To seek a tradeoff between delay and utilization, the recommended REF_QS is 20%~60% of the buffer size.

Experiment 3 – Time-varying RTT

In order to investigate the *S/U* MSP controller performance when the round trip time (RTT) is time-varying, we now change the round trip propagation delays (RTPDs) of all connections with the time: from 0 to 100 seconds, all RTPDs are set as 100ms; at the time of 100 seconds, they are increased to 270ms and are kept at this value until the simulation time of 200 seconds, when all RTPDs are decreased to 50ms.

The instantaneous queue sizes under the RED, the PI controller and the *S/U* MSP controller are given in Fig. 5.15. In the first simulation period (0~100 sec), all the four controllers can stabilize the queue size, but the settling times are different: 30 seconds (RED), 25 seconds

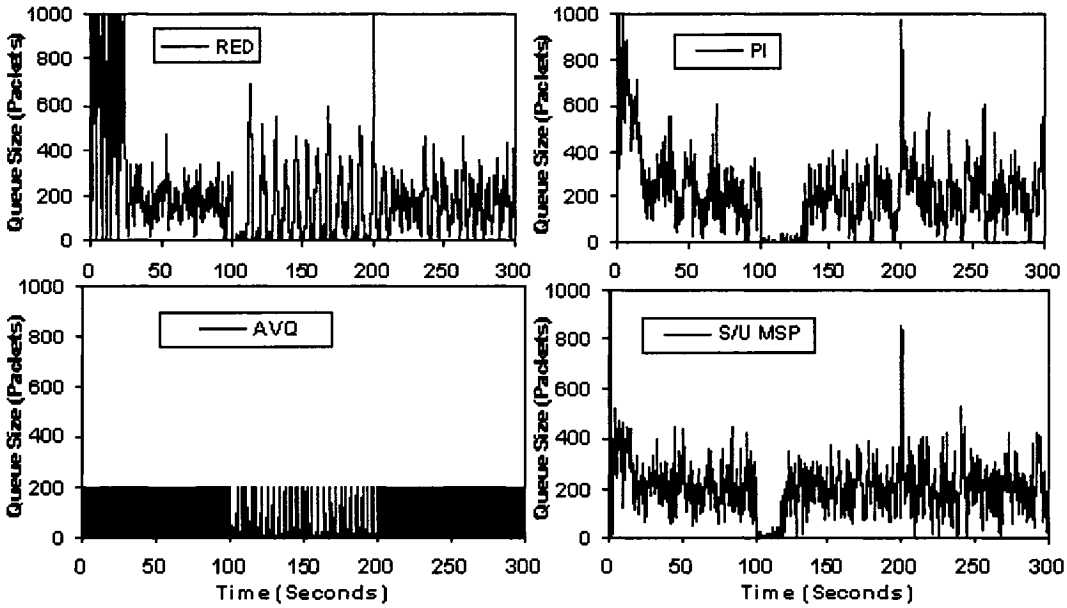

Fig. 5.15: Instantaneous queue size vs. Time: comparison of the RED, the PI, the AVQ and the *S/U* MSP controllers in the case of time-varying RTTs, RTPD varies from 100ms (0 - 100 sec), 270ms (100 - 200 sec) to 50ms (200 - 300 sec)

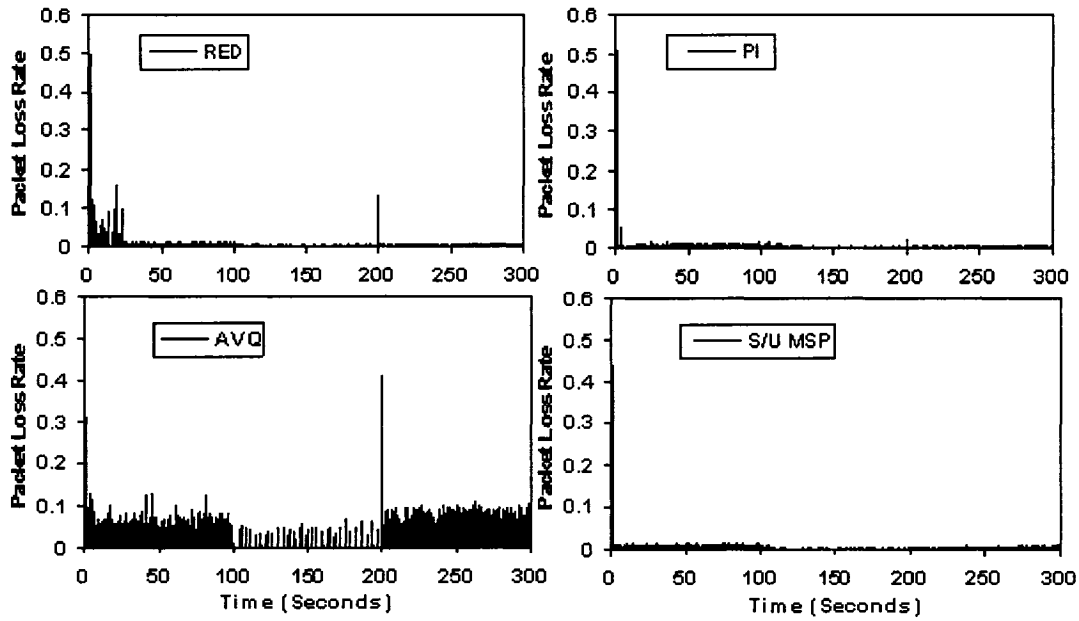

Fig. 5.16: Packet loss rate vs. Time: comparison of the RED, the PI, the AVQ and the *S/U* MSP controllers in the case of time-varying RTTs, RTPD varies from 100ms (0 - 100 sec), 270ms (100 - 200 sec) to 50ms (200 - 300 sec)

(PI), 0 second (AVQ), and 20 seconds (*S/U* MSP). Although settling time for the AVQ is short, it is at the cost of large packet loss rate, which is confirmed from the packet loss rate

comparison diagram Fig. 5.16. In the second simulation period (100~200 sec), the queue size under the RED shows larger oscillations than that under the PI controller, the AVQ and the *S/U* MSP controller. The settling time under the PI controller (34 sec) is larger than that under the *S/U* MSP (20sec), which in turn is larger than that under the AVQ (7 seconds). When it comes to the third period (200~300sec), all four controllers can quickly stabilize the queue size, and the settling time under the PI controller is larger than that under the RED, the AVQ and the *S/U* MSP. But the AVQ shows large packet loss rate again (refer to Fig. 5.16) in the third simulation period. And the AVQ does not have the capability to clamp the queue size to the REF_QS of 200 packets in all the three simulation periods.

The packet loss rate comparison for the four controllers is presented in Fig. 5.16. It shows that, after the packet loss rates become steady-state, the variations in packet loss rate for the RED, the PI and the *S/U* MSP are similar: with the range from 0% to 1%, but the AVQ shows a significant larger range from 0% to 10% during the simulation period of 0~100 sec and 200~300 sec.

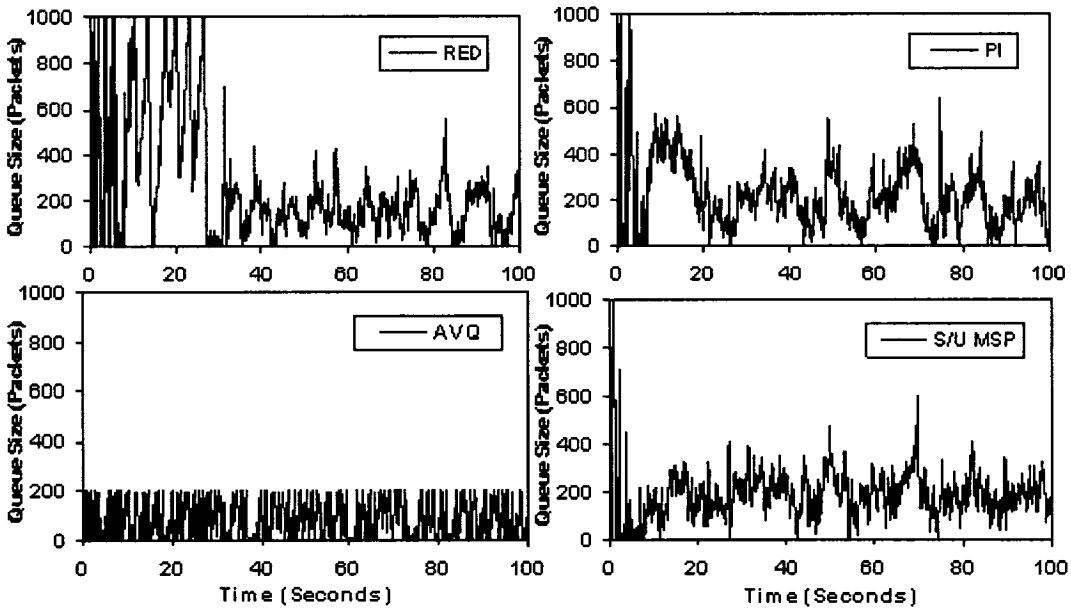

Fig. 5.17: Instantaneous queue size vs. Time: comparison of the RED, the PI, the AVQ and the *S/U* MSP controllers in the case of different RTTs for different connections

Experiment 4 - Different RTT for different connections

In this experiment, different connections have different RTTs. Totally we have 100 ftp connections and 20 http connections. The RTPDs are set as follows. For the ftp sources, the RTPDs are 100ms, 200ms, 300ms and 400ms for the connection 1~25, connection 26~50, connection 51~75 and connection 76~100 respectively, and for the http sources, the RTPDs

are set to 100ms and 400ms for connection 1~10 and connection 11~20 respectively.

The instantaneous queue size is given in Fig. 5.17. It is shown that all the four controllers can stabilize the queue size, but with different settling times. The RED takes 30 seconds to stabilize the queues size, whereas the PI controller and the *S/U* MSP controller takes 7 seconds to clamp the queue size to the steady state, and the AVQ takes almost zero second to clamp the queue size to a low value of 100 packets, which is far from the REF_QS of 200 packets. The queue size oscillations under the PI controller are larger than those under the *S/U* MSP controller and the AVQ.

Table 5.1 Round trip propagation delay (RTPD) configuration for Section 5.5 Experiment 5

Simulation time t (sec)	$0 < t \leq 50$	$50 < t \leq 100$	$100 < t \leq 150$
Connection ID	RTPD (ms)	RTPD (ms)	RTPD (ms)
ftp 1~25	50	100	25
ftp 26~50	100	200	50
ftp 51~75	150	300	75
ftp 76~100	200	400	100
http 1~10	50	100	25
http 11~20	200	400	100

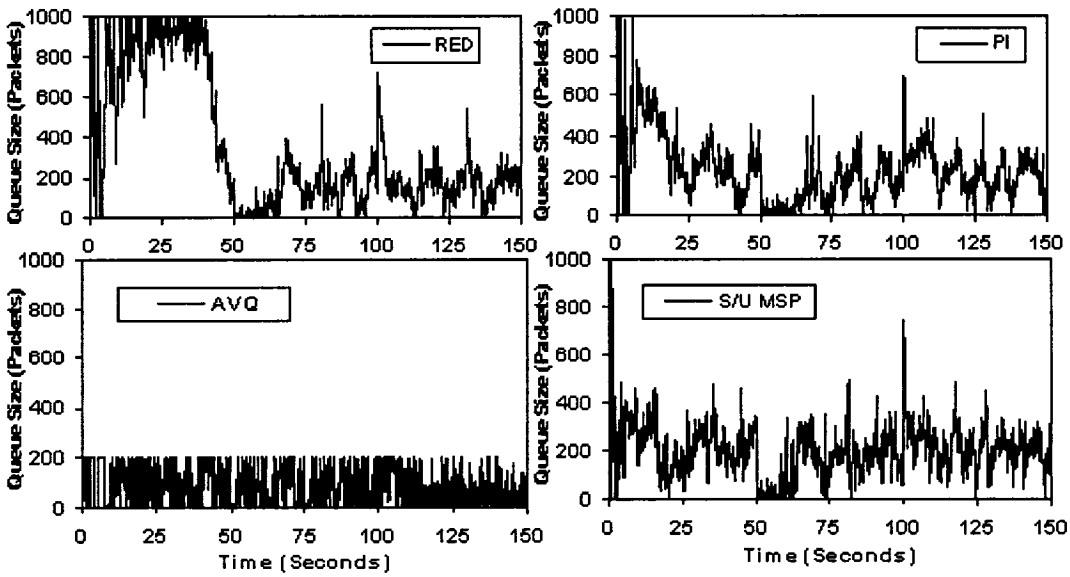

Fig. 5.18: Instantaneous queue size vs. Time: comparison of the RED, the PI, the AVQ and the *S/U* MSP controllers in the case of different time-varying RTTs for different connections

Experiment 5 - Different time-varying RTT for different connections

In this experiment, we use different time-varying RTTs for different connections. Namely the RTPDs are different for different connections and the RTPDs are also changed with time. Table 5.1 presents the RTPD configuration.

Fig. 5.18 shows the instantaneous queue size for the four controllers under the above configuration. In the first simulation period (0~50 sec), the RED has difficulty to stabilize the queue size, whereas the PI controller, the AVQ and the *S/U* MSP controller can both stabilize the queue size. But the response of the PI controller is slower than that of the *S/U* MSP controller, which in turn is slower than that of the AVQ. But the AVQ cannot clamp the queue size to the REF_QS of 200 packets. In the second (50~100 sec) simulation period, all the four controllers can stabilize the queue size, but with different settling time: 16 seconds for the RED, 14 seconds for the PI controller, 0 second for the AVQ, and 11 seconds for the *S/U* MSP controller. In the third (100~150 sec) simulation period, all the four controllers can drive the queue size to the steady state with similar settling time.

5.6 Concluding Remarks

This chapter exploits the modern $\mathcal{H}_\infty$ optimal control theory to propose the *S/U* MSP controller and the *S/T/U* MSP controller design for AQM routers. By forming the *S/U* MSP and the *S/T/U* MSP in the general control configuration, the controllers can be synthesized using a state-space solution. Through the usage of the loop shaping technique, the closed-loop dynamics of the system can be explicitly bounded by the weight functions. The guidelines for the choices of weight functions are determined through theoretical analysis and/or simulations. Extensive simulations including in the case of different time-varying RTTs for different connections have verified the effectiveness of the controllers. The work demonstrates the successful employment of the modern $\mathcal{H}_\infty$ optimal control theory to the AQM design.

As mentioned at the beginning of Section 5.5, the simulation results show that the *S/U* MSP controller (5.4) and the *S/T/U* MSP controller (5.6) exhibit very similar performance. There is no significant difference between them in terms of the performance. Actually we know that if difference does exist, it should come from the selection of weight W_T (5.5) in the *S/T/U* MSP controller design. In our example, the selection of W_T in the *S/T/U* MSP controller design in Section 5.4 happens to give a Bode diagram of T similar to that under the *S/U* MSP controller design in Section 5.2, which is verified by comparing Fig. 5.11 with Fig. 5.5. The similarity of the T Bode diagrams explains that there is no significant performance difference under the *S/U* MSP controller (5.4) and the *S/T/U* MSP controller (5.6).

After we design the controller using the *S/U* MSP method, if the resulted *T* Bode diagram shows sufficient roll-off, as does in our example, then we do not need to use the *S/T/U* MSP method, because of two reasons: (a) the *S/T/U* MSP controller is one order higher than the *S/U* MSP controller, as can be seen from (5.4) and (5.6); (b) in the *S/T/U* MSP controller design, we have the additional work to determine the parameters of the weight W_T.

On the other hand, if the Bode diagram of the transfer function *T* under the *S/U* MSP controller shows no or insufficient roll-off, we still need the *S/T/U* MSP controller. Furthermore, when we need to consider the robustness against uncertainties analytically, the *S/T/U* MSP is far more related, as we can see in the next chapter.

With the *S/U* MSP controller design and the *S/T/U* MSP controller design proposed in this chapter, we cannot consider explicitly the uncertainties of the TCP/AQM system, and the robust stability (RS) and robust performance (RP) of the system cannot be analytically evaluated. Therefore we shall consider another robust control approach – robust μ analysis in the next chapter.

Chapter 6 AQM Design Based on Robust μ Analysis

Unlike the $\mathcal{H}_\infty$ controller considered in the last chapter, we shall consider explicitly here the uncertainties in the form of the uncertainty weight. We first summarize the control technique of the robust μ analysis. We then develop the design of the MU_OPT controller, perform the analysis on the nominal performance (NP), the robust stability (RS) and the robust performance (RP) of the system, and finally evaluate the performance evaluations of the MU_OPT controller through extensive simulations.

We have summarized in Appendix C some preliminaries on the robust μ analysis, including the uncertainty modeling and the analysis of NP, RS and NP using the μ analysis technique. More details can be found in [SkPo01].

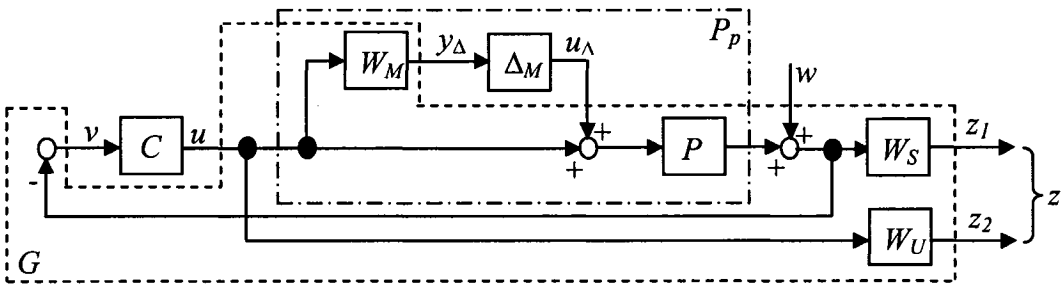

Fig. 6.1: Feedback system with the multiplicative uncertainty

6.1 The Roust μ Analysis Model and Method

Fig. 6.1 illustrates the model configuration of our MU_OPT controller, where C is the controller we are trying to synthesize, and the symbols w, z, v, u, G and P have been described in Section 5.1. Recall that we did not consider the system uncertainties in Chapter 5, but such uncertainties do exist in a practical TCP/AQM system. Therefore, we introduce in Fig. 6.1 the perturbed plant model P_p, which consists of the nominal plant P, the lumped multiplicative uncertainty weight W_M, and the scaled multiplicative uncertainty Δ_M. Here, Δ_M is a stable transfer function with a magnitude less than or equal to one at any frequencies.

We want to make our TCP/AQM control system robust in the sense that it is insensitive to differences between the actual system and the design model of the AQM controller. The differences are referred to as model uncertainty. The model uncertainty may have many sources. Some of these are (i) the network parameters such as N, R_0 and C of the

linear TCP/AQM model are only known approximately; (ii) the network parameters such as N, R_0 and C may vary due to changes in the operating conditions; (iii) at high frequencies even the structure and the model order is unknown; (iv) even when a very detailed model is available we may choose to work with a simpler (low-order) nominal model and represent the neglected dynamics as "uncertainty". Generally we group the above sources into two main classes:

1) Parametric uncertainty: the structure and the order of the TCP/AQM model are known, but the network parameters such as N, R_0 and C are uncertain.
2) Neglected and unmodeled dynamic uncertainty: the TCP/AQM model is in error because of missing dynamics, usually at high frequencies, either through deliberate neglect or because of a lack of understanding of the physical process. For example, the TCP/AQM model ignores the TCP timeout mechanism.

In our analysis, we combine the parametric uncertainty and the neglected / unmodeled uncertainty into one single lumped perturbation P_p as shown in Fig. 6.1,

$$P_p(s) = P(s)(1 + W_M(s)\Delta_M(s)) \quad , \tag{6.1}$$

where $\Delta_M(s)$ satisfies: $|\Delta_M(j\omega)| \le 1 \quad \forall \omega$.

As discussed in Appendix C, the NS (Nominal Stability), the NP, the RS, and the RP of a system can be described by condition (C.7) and three μ–conditions (C.8)-(C.10). In our TCP/AQM system, these conditions become

$$\begin{aligned} &\text{NS} \Leftrightarrow \text{closed-loop system is nominally stable;} \\ &\text{NP} \Leftrightarrow |W_S S| < 1, \ \forall \omega \text{, and under condition NS;} \\ &\text{RS} \Leftrightarrow |W_M T| < 1, \ \forall \omega \text{, and under condition NS;} \\ &\text{RP} \Leftrightarrow |W_S S| + |W_M T| < 1, \ \forall \omega \text{, and under condition NS.} \end{aligned} \tag{6.2}$$

From (6.2), it can be inferred that the robust performance optimization involves minimizing the peak value of the robust performance index $\mu(N) = |W_S S| + |W_M T|$, where N is the lower LFT[11] (linear fractional transformation) of G and C (refer to Appendix C for details). Unfortunately at present there is no direct method to synthesize a μ-optimal controller. A closely related problem, which is easier to solve both mathematically and numerically, is to

[11] N is also used to denote the number of long-lived TCP connections, and this can be easily determined by the context.

minimize the peak value ($\mathcal{H}_\infty$ norm) of the mixed sensitivity matrix [SkPo01]

$$N_{mix} = \begin{bmatrix} W_S S \\ W_M T \end{bmatrix} .$$

In the case of the TCP/AQM design, since we need to bound the plant input δp (which is the variation of the packet drop probability), we need to further approximate the robust performance optimization problem by introducing the term $W_U U$ in N_{mix},

$$N_{mix} = \begin{bmatrix} W_S S \\ W_M T \\ W_U U \end{bmatrix} ,$$

where W_U and U are the input sensitivity weight and the input sensitivity function respectively. In other words, we can now minimize $\mu(N)$ by solving the $\mathcal{H}_\infty$ $S/T/U$ mixed sensitivity problem (MSP). The solution for the $S/T/U$ MSP can be found in [DoGl89] and is available in the μ-toolbox of MATLAB [MATL02]. In essence, there are 2 steps in the solution: 1) Solving the $\mathcal{H}_\infty$ sub-optimal control problem: For a specified real number γ a stabilizing controller needs to be found for which $\|N\|_\infty < \gamma$. 2) Reducing γ iteratively: The algorithm in 1) is used iteratively, reducing γ until the minimum is reached within a given tolerance and therefore the optimal controller is obtained.

6.2 Design of the MU_OPT Controller

In this section, we would present the procedure for the design of μ-optimal controller for TCP/AQM system. First the uncertainty modeling of the TCP/AQM system is discussed. Then we approximate the μ-optimal controller (the MU_OPT controller) by solving the $\mathcal{H}_\infty$ $S/T/U$ MSP with the uncertainty modeling. Finally the NP, RS and RP of the system under the synthesized controller are analyzed quantitatively.

6.2.1 Determining the Uncertainty Weight for the TCP/AQM System

In the robust controller design, the first step is to determine the uncertainty weight $W_M(s)$ in (6.1). The following describes a graphical technique to determine W_M. First we discuss the choice of W_M from the *parametric uncertainty* point of view. Fig. 6.1 shows that,

$$\Delta_M(j\omega) = \frac{\frac{P_p(j\omega) - P(j\omega)}{P(j\omega)}}{W_M(j\omega)} \quad ,$$

where Δ_M, P_p, P, W_M are defined in Section 6.1. To satisfy $|\Delta_M(j\omega)| \le 1 \quad \forall \omega$, which is the definition of Δ_M, we have,

$$|W_M(j\omega)| \ge \left| \frac{P_p(j\omega) - P(j\omega)}{P(j\omega)} \right| \qquad \forall \omega \quad .$$

This means that the magnitude of $W_M(j\omega)$ must be greater than the magnitude of the plant relative error $\left| \frac{P_p(j\omega) - P(j\omega)}{P(j\omega)} \right|$. Therefore we can choose $W_M(s)$ as follows. First we determine n uncertain parameters (e.g., N, R_0 and C in the TCP/AQM system) in the plant model and assume that each uncertain parameter is bounded within some region. For each parameter, we choose three representative values within their respective regions so that we could get 3^n combinations[12] for the perturbed plant $P_p(s)$. Then we could get $W_M(s)$ using a graphical method: we draw the magnitude diagram of the 3^n combinations of the relative errors; then from this diagram we can fit a $W_M(s)$ so that the magnitude curve of $W_M(s)$ is above all those 3^n relative error curves.

Next we discuss the choice of $W_M(s)$ from the *unmodeled dynamics uncertainty* point of view. To represent unmodeled dynamics we usually choose the multiplicative weight $W_M(s)$ in the form [SkPo01]

$$W_M(s) = \frac{Js + r_0}{(J / r_\infty)s + 1} \quad , \tag{6.3}$$

where r_0 is the relative uncertainty at steady state (low frequency); $1/J$ is approximately the frequency at which the relative uncertainty reaches 100%; and r_∞ is the magnitude of the weight at high frequency.

From the previous description, $W_M(s)$ can be determined in the following procedure:

1) *Choose the nominal model P(s) in a simplified form, e.g. a low-order, delay-free model;*
2) *Determine n uncertain parameters in the plant model and assume that each uncertain parameter is bounded within some region;*

[12] This is not, in general, guarantee to yield the worst case for the relative error since the worst case may be at the interior of the intervals. But as justified by the simulations, this approximation method is accurate enough for the TCP/AQM controller design.

3) *Choose three representative values for each parameter within their respective regions, therefore 3^n combinations of the perturbed plant $P_p(s)$ and their respective relative error are obtained;*
4) *Draw the magnitude curves of the 3^n relative errors in one diagram. Choose $W_M(s)$ in the form of (6.3), determine appropriate values for J, r_0 and r_∞ to make sure the magnitude curve of $W_M(s)$ is above all the magnitude curves of the 3^n relative errors.*

Please note that we only need to perform the above procedure in the design phase to determine W_M. After we finish our design, namely, after the controller has been determined, the controller is then implemented using the method described in Section 2.3.3. Now we apply the above procedure to the uncertainty modeling of the TCP/AQM system through an example. From Section 2.2, the plant transfer function of the TCP/AQM control system is given by

$$P_p(s) = \frac{\frac{C^2}{2N} e^{-sR_0}}{(s+\frac{2N}{R_0^2 C})(s+\frac{1}{R_0})} .$$

Here we use the subscript 'p' to mean 'perturbed', which means the plant is perturbed because of the uncertainties in the plant parameters N, R_0 and C. For the nominal plant $P(s)$, we use a delay-free model

$$P(s) = \frac{\frac{\overline{C}^2}{2\overline{N}}}{(s+\frac{2\overline{N}}{\overline{R}_0^2 \overline{C}})(s+\frac{1}{\overline{R}_0})} ,$$

where $\overline{N}$, $\overline{R}_0$ and $\overline{C}$ are the nominal values for N, R_0 and C respectively. Through using the first-order Padé approximation [Wong92] for the delay term e^{-sR_0} in $P_p(s)$, we get the plant relative error

$$\left|\frac{P_p(s)-P(s)}{P(s)}\right| = \left|\frac{\frac{C^2}{2N}(1-\frac{R_0}{2}s)(s+\frac{2\overline{N}}{\overline{R}_0^2\overline{C}})(s+\frac{1}{\overline{R}_0}) - \frac{\overline{C}^2}{2\overline{N}}(s+\frac{2N}{R_0^2 C})(s+\frac{1}{R_0})(1+\frac{R_0}{2}s)}{\frac{\overline{C}^2}{2\overline{N}}(s+\frac{2N}{R_0^2 C})(s+\frac{1}{R_0})(1+\frac{R_0}{2}s)}\right| . \quad (6.4)$$

Let us take an example with the following parameter values, a) the nominal values for N, R_0 and C are: $\overline{N}$=100, $\overline{R}_0$=0.25 sec, $\overline{C}$=9708 packet/sec; b) the parameters N, R_0 and C

all exhibit ±8% uncertainty around their nominal values, namely $N \in (92, 108)$, $R_0 \in (0.23, 0.27)$, $C \in (8931, 10485)$. We consider three values for each of the three parameters: 92, 100 and 108 for N; 0.23, 0.25 and 0.27 for R_0; 8931, 9708 and 10485 for C. The corresponding relative errors $|(P_p(s)-P(s))/P(s)|$ are shown as function of frequency for the $3^3=27$ resulting P_p's in Fig. 6.2.

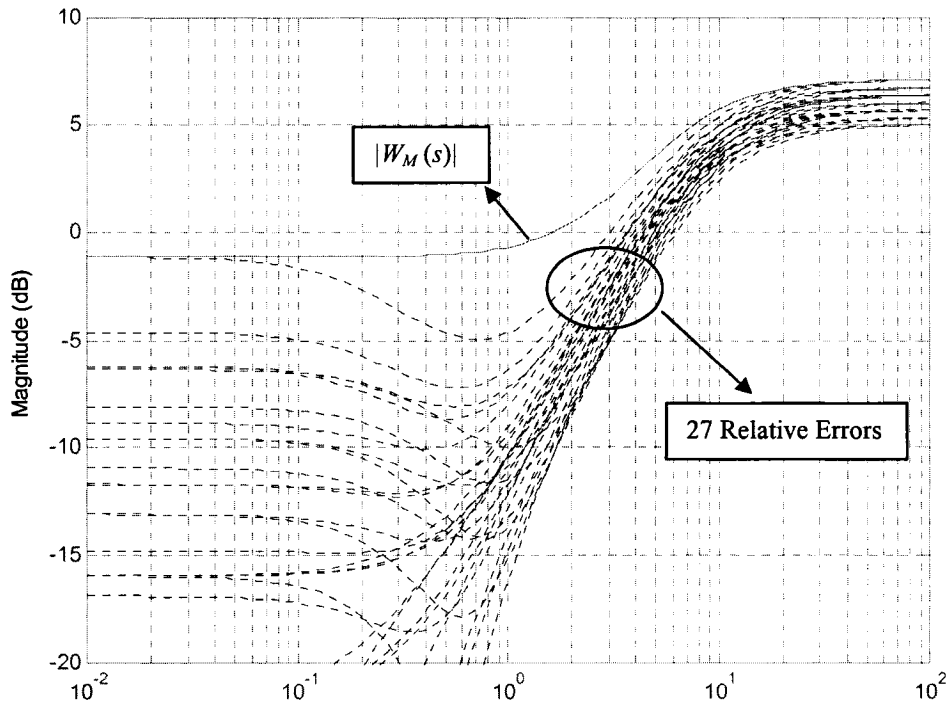

Fig. 6.2: Relative errors for 27 combinations of N, R_0 and C of delay-free nominal plant (dotted lines) against the uncertainty weight $|W_M(s)|$ (the solid line)

We choose $W_M(s)$ in the form of (6.3), and the curve for $W_M(s)$ must at each frequency lie above all the dotted lines in Fig. 6.2. From Fig. 6.2, we observe that $W_M(s)$ should be 0.874 (-1.17dB) at low frequencies and 2.262 (7.09dB) at high frequencies, and the up-most dotted line cross 100% (0dB) at 3.02 rad/sec. Therefore we have $r_0=0.874$, $r_\infty=2.262$ and $T_1=1/3.02=0.331$sec. Namely, we can choose the following uncertainty weight,

$$W_M(s) = \frac{0.331s + 0.874}{(0.331/2.262)s + 1} \quad . \tag{6.5}$$

As can be seen from the solid line in Fig. 6.2, this weight does lie above all relative error curves.

6.2.2 The MU_OPT Controller Design

As stated in Section 6.1, the μ optimal controller can be approximately obtained by solving the $\mathcal{H}_\infty$ $S/T/U$ MSP

$$C_{\text{opt}} = \underset{C \text{ stabilizes } P}{\text{minimize}} \left\| \begin{bmatrix} W_S S \\ W_M T \\ W_U U \end{bmatrix} \right\|_\infty .$$

where W_S, W_M and W_U are the performance weight, the uncertainty weight and the input sensitivity weight, respectively. Details of the approximation can be found in Appendix C.

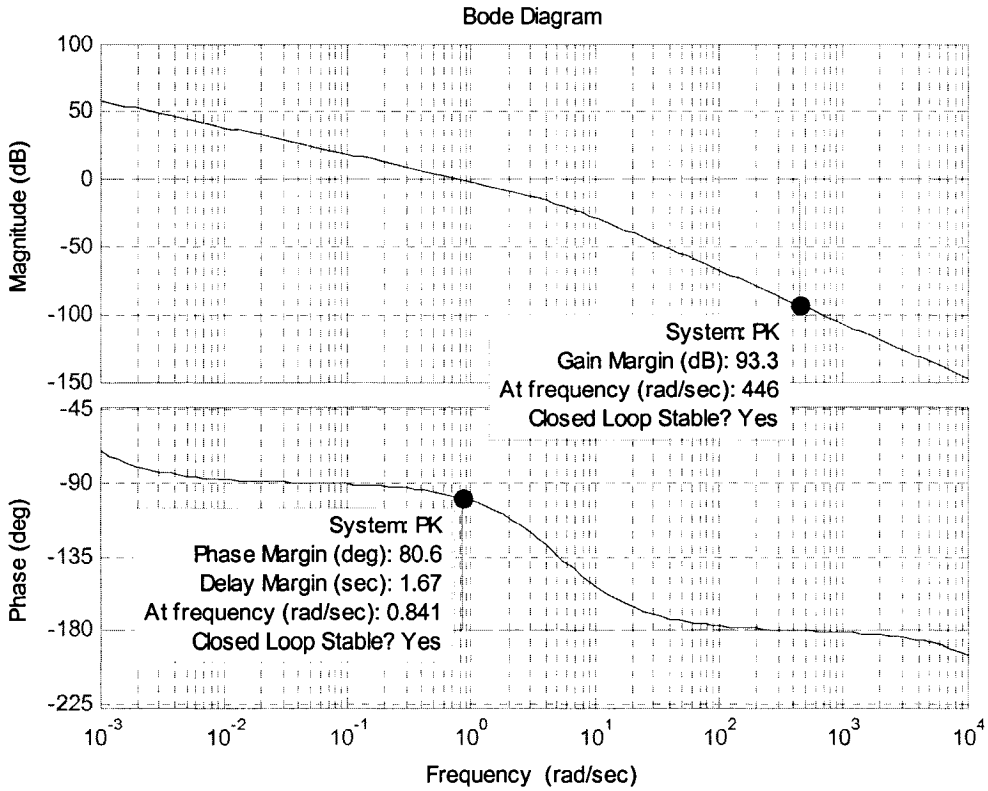

Fig. 6.3: Bode diagram of the open-loop transfer function PC (system PK)

As discussed in Section 5.2, usually we choose the performance weight W_S in the form

$$W_S = \frac{s/M + \omega_B}{s + \omega_B A} ,$$

where M, ω_B, and A are defined in Section 5.2. From Section 5.3.5 we choose $M = 1.5$, $\omega_B = 3.5$, $A = 10^{-4}$. As discussed in Section 6.2.1, W_M should be chosen in the form of (6.5). In order to limit the plant input, we choose $W_U = 4.75 \times 10^5$ from the guidelines presented in

Section 5.3.5. Since P(s), W_S(s), W_M(s) and W_U have been determined, with the μ-toolbox in MATLAB, we could get the following MU_OPT controller

$$C(s) = \frac{(0.250111s^3 + 2.79208s^2 + 7.72998s + 2.25358)\times 10^{-6}}{6.94729\times 10^{-7}s^4 + 0.027304s^3 + 0.32479s^2 + 0.94539s + 3.30847\times 10^{-4}} \quad . \tag{6.6}$$

The optimal $\mathcal{H}_\infty$ norm value γ_{min} is equal to 4.4032. Fig. 6.3 gives the Bode diagram of the open loop transfer function PC, and here P is the nominal plant. From the figure, we conclude that the Phase Margin is 80.6°, and the Gain Margin is 93.3 dB. Therefore the system is closed-loop stable under the nominal condition, namely the system satisfies the NS (Nominal Stability) condition.

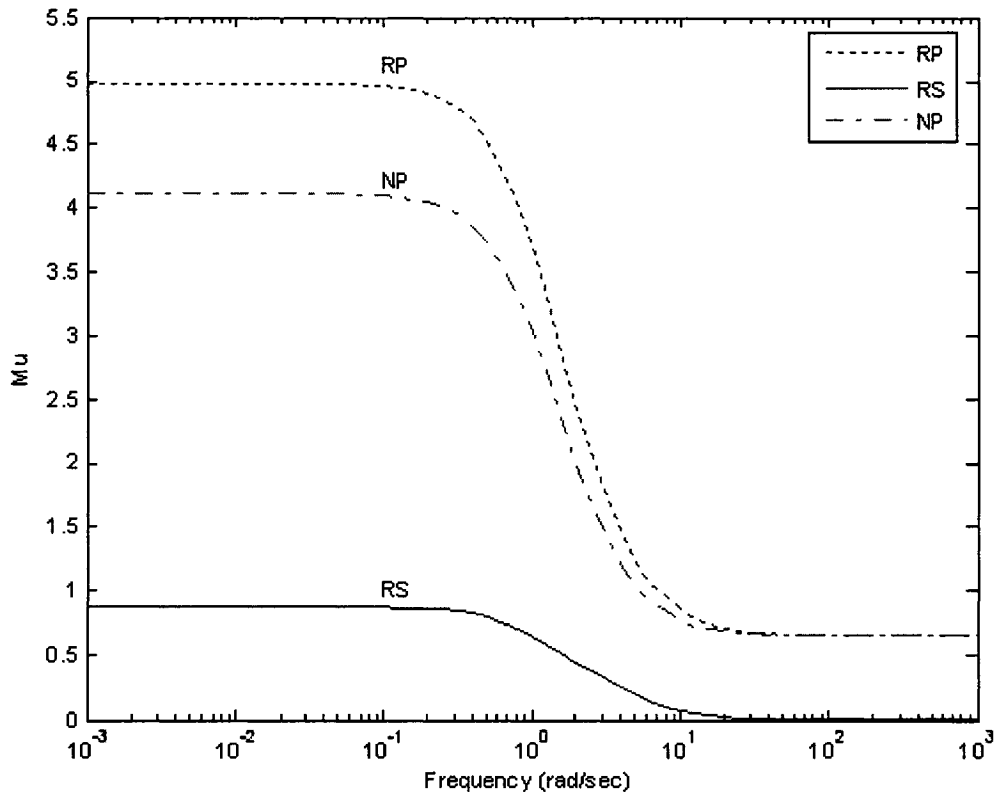

Fig. 6.4: RP, NP and RS μ curves for the TCP/AQM system with the MU_OPT controller

6.2.3 Analysis of the NP, the RS and the RP of the TCP/AQM system

As discussed and summarized in Appendix C.3, it turns out that the NP, RS and RP can be achieved as long as their corresponding μ value is less than one. For the case when the μ value is greater than one, the NP, RS or RP cannot be achieved, but it does indicate "how far" the NP, RS or RP can be satisfied. For example, if the structured singular value

$\mu_{\Delta_M}(N_{11}) < r$, but r>1, it means we have to decrease the uncertainty by a factor of $1/r$ before the worst-case uncertainty yields instability, i.e., we can tolerate about $1/r$ uncertainty before we get instability. Similarly, if r<1, it means we can increase the uncertainty by a factor of $1/r$ before instability occurs. Therefore, those μ values in (C.8)-(C.10) are actually characteristic indices for NP, RS and RP. Consequently, when we are performing the μ–analysis for a system with a given controller, we usually draw those μ curves in a diagram and label them as NP, RS or RP. Section 6.2.2 has already demonstrated that the TCP/AQM system considered in this thesis with the controller designed by (6.6) satisfies the NS condition. In this subsection, we further investigate the NP, RS and RP of the system by drawing their corresponding μ curves.

Fig. 6.4 gives the μ curves for the TCP/AQM system with the MU_OPT controller (6.6). It shows that at all frequencies μ values for RS are less than one, which means robust stability of the system is satisfied at all frequencies for all plants within the uncertainty ranges described in Section 6.2.1. Actually the maximum μ value for RS is 0.8736, which means even the uncertainty is increased by a factor of 1/0.8736=1.145, the system can still maintain stability. However, NP and RP cannot be satisfied because their maximum μ values are 4.1061 and 4.9797 respectively, and both values are greater than one.

We can verify our claim that "even the uncertainty is increased by a factor of 1/0.8736=1.145, the system can still maintain stability" through simulations. We know that when the parameters N, R_0, C deviate from their nominal values, uncertainties occur. Therefore in order to verify the statement, we could change N alone, R_0 alone, C alone or change them simultaneously and check the ratio of the plant relative error (RE) divided by the uncertainty weight W_M (we denote this ratio as RE/WM). If the ratio is less than 1.145 (1.176dB), the queue size should still be stable. On the contrary, if the uncertainties are so large that the ratio is greater than 1.176dB, the queue size cannot be stabilized. From this we can show how we calculate theoretical stability ranges for N, R_0, or C by varying one parameter alone while keeping the other two parameters fixed at their nominal values. All calculated theoretical stability ranges will further be verified through simulations in Section 6.3.

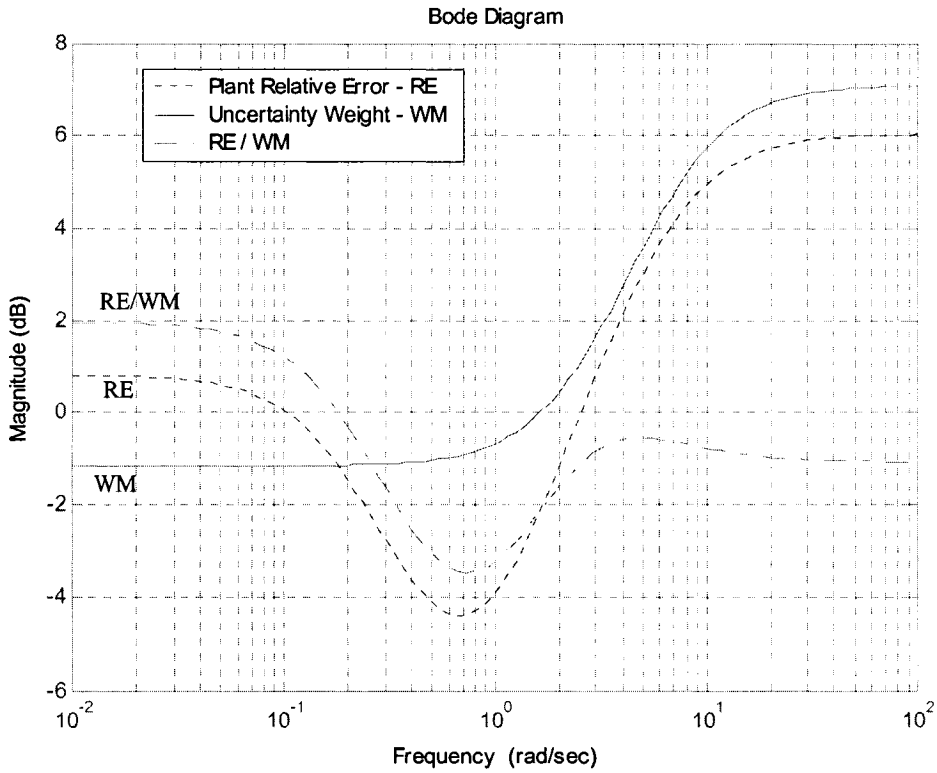

Fig. 6.5: Bode diagrams of the Plant Relative Error (RE), the Uncertainty Weight (WM) and the ratio between them (RE/WM) when R_0 = 0.32s (N and C are at nominal values)

First we calculate the theoretical stability range for R_0 when varying R_0 alone (the other two parameters N and C are fixed at their nominal values). From (6.4) and (6.5), we could draw Bode diagrams of the plant relative error (RE), the uncertainty weight W_M and the ratio RE/WM. As analyzed in the previous paragraph, it is required that the maximum RE/WM ratio should be less than 1.176dB to ensure stability. We have plotted many Bode diagrams of RE/WM with different values of R_0. Two critical results are as follows: when R_0=0.315s, we have $(RE/WM)_{max}$ = 1.1649dB; and when R_0 = 0.32s, we obtain $(RE/WM)_{max}$ = 1.9663dB. We therefore determine the approximate theoretical stability range for R_0 is $R_0 \leq$ 0.315s. Fig. 6.5 gives the Bode diagrams of RE, WM and RE/WM when R_0 is equal to 0.32s. It is shown that the maximum RE/WM ratio occurs at a low frequency and is equal to 1.9663dB when R_0 is equal to 0.32s.

Similarly we could also determine the theoretical stability range for N when varying N alone. From the Bode diagrams generated using (6.4) and (6.5), we have the following two critical results when varying N: when N=71, we have $(RE/WM)_{max}$ = 1.0213dB; when N = 70, we obtain $(RE/WM)_{max}$ = 1.5110 dB. We therefore determine the approximate theoretical

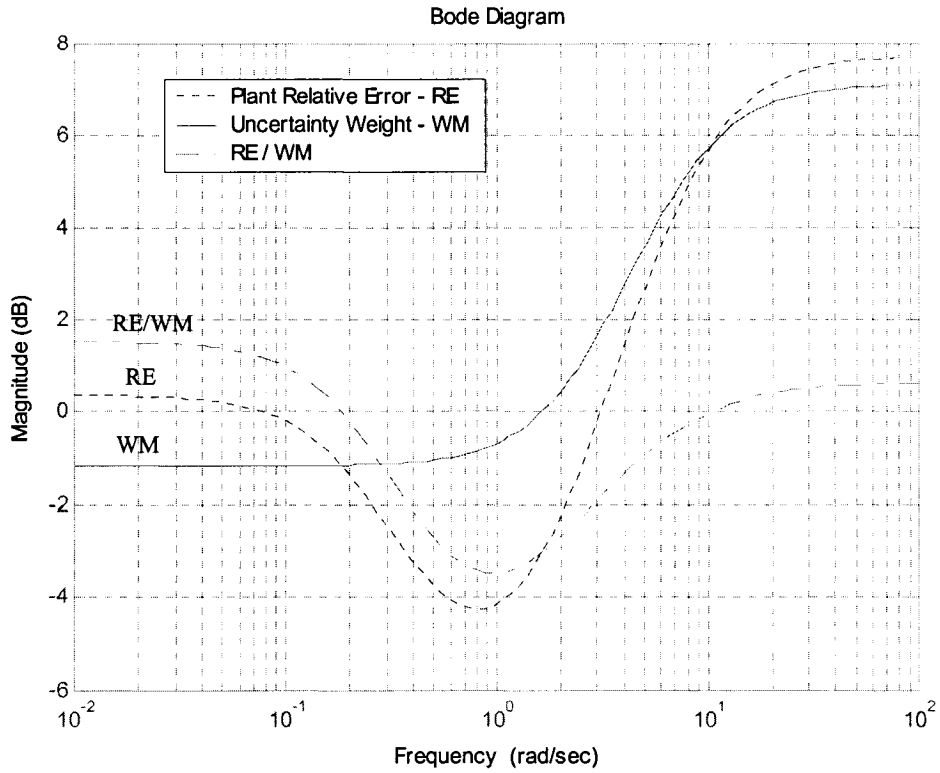

Fig. 6.6: Bode diagrams of the Plant Relative Error (RE), the Uncertainty Weight (WM) and the ratio between them (RE/WM) when $N = 70$ (R_0 and C are at nominal values)

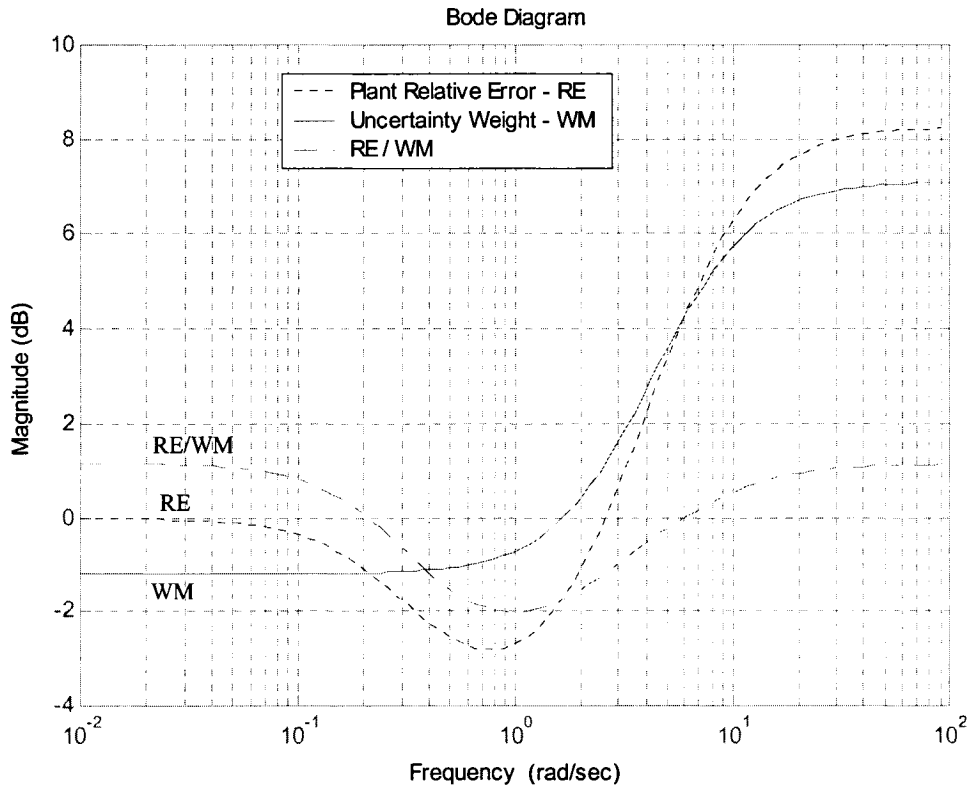

Fig. 6.7: Bode diagrams of the Plant Relative Error (RE), the Uncertainty Weight (WM) and the ratio between them (RE/WM) when $C = 12234$ pkt/sec (R_0 and N are at nominal values)

stability range for N is $N \geq 71$. Fig. 6.6 gives the Bode diagrams of RE, WM and RE/WM when N is equal to 70. It is shown that the maximum RE/WM ratio occurs at a low frequency and is equal to 1.5110dB when N is equal to 70.

The same procedure can be done to obtain the theoretical stability range for C when varying C alone. From the two critical Bode diagrams: when C=12233 packet/sec, we have $(RE/WM)_{max}$ = 1.1731dB; and when C = 12234 packet/sec, we obtain $(RE/WM)_{max}$ = 1.1774 dB. Thus the approximate theoretical stability range for C is determined as $C \leq 12233$ packet/sec. Fig. 6.7 presents the Bode diagrams of RE, WM and RE/WM when C is equal to 12234 packet/sec. It is shown that the maximum RE/WM ratio occurs at a low frequency and is equal to 1.1774 dB when C is equal to 12234 packet/sec.

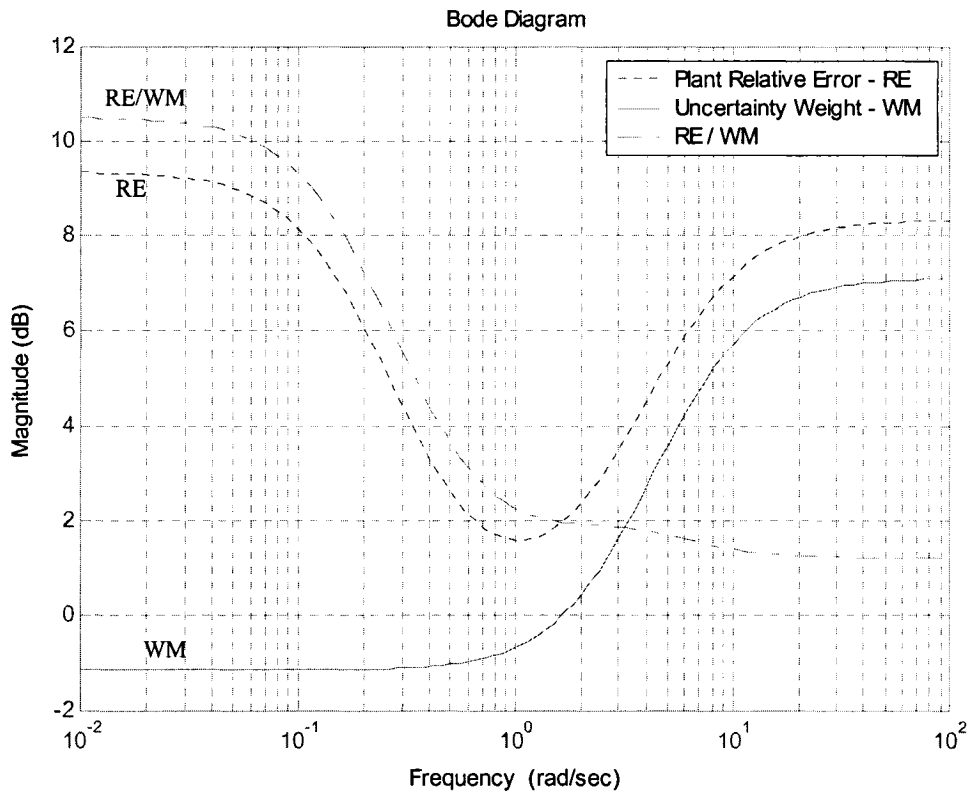

Fig. 6.8: Bode diagrams of the Plant Relative Error (RE), the Uncertainty Weight (WM) and the ratio between them (RE/WM) when N=80, R_0=0.3s and C = 11000 packet/sec

On the other hand, if R_0, N, C all deviate from their nominal values, we could also theoretically determine whether the system is stable or not for any combination of R_0, N and C by drawing the Bode diagram of RE/WM. For example, we consider the following three combinations of network parameters: (a) N=150, R_0=0.32s, C=8500 packet/sec; (b) N=80,

R_0=0.3s, C=11000 packet/sec; (c) N=100, R_0=0.12s, C=10000 packet/sec. From the Bode diagrams of RE/WM, we find that $(RE/WM)_{max}$ is equal to −7.3646dB, 10.4861dB and 0.0506dB for the cases of (a), (b) and (c) respectively. Since the stable condition is that $(RE/WM)_{max}$ should be less than 1.176dB, therefore we deduce that in the case of (b), the system is unstable, whereas in the case of (a) and (c) the system is stable. This will further be verified through the simulation in Section 6.3 Experiment 7. Fig. 6.8 gives the Bode diagrams of RE, WM and RE/WM in the case of (b). It is shown that the maximum RE/WM ratio occurs at a low frequency and is equal to 10.4861dB.

6.3 Performance Evaluation for the MU_OPT Controller

The MU_OPT controller (6.6) has been verified via simulations using OPNET Modeler [OPNE01]. The simulation setup has been described in Section 2.3.1, and we use the same parameter settings as in Section 3.2, i.e. N=100, R_0=0.25 second and C=9708 packets/second as the nominal values, and in some experiments, we may vary those values in order to investigate the robustness against uncertainties. By default the REF_QS is set as 200 packets.

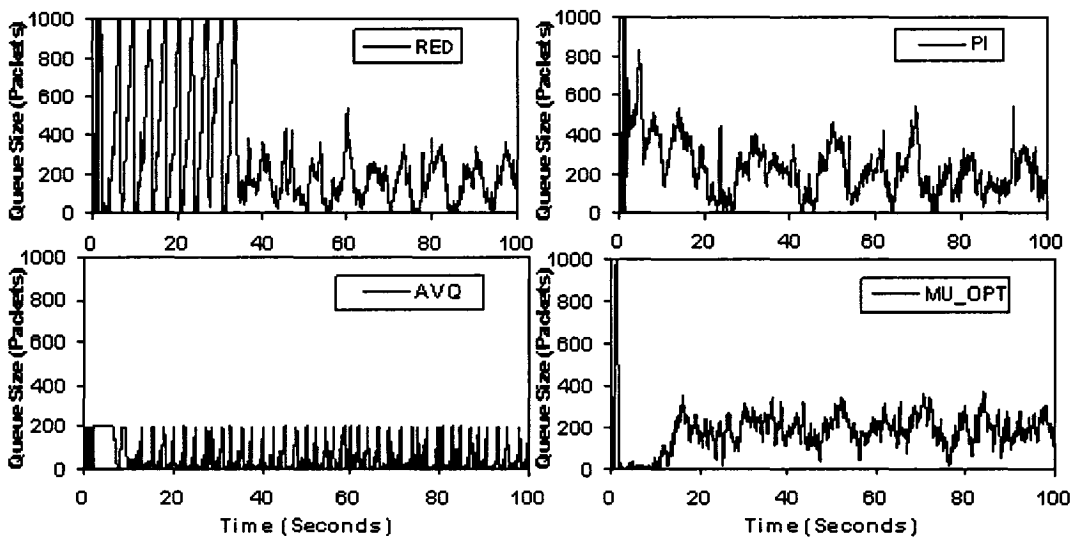

Fig. 6.9: Instantaneous queue size vs. Time: comparison of the RED, the PI, the AVQ and the MU_OPT controllers

Experiment 1 – Comparison of RED, PI, AVQ and MU_OPT

This experiment compares the performance of the RED, the PI controller, the AVQ and the MU_OPT controller. Fig. 6.9 gives the instantaneous queue size evolution comparison under the four controllers in the nominal condition (i.e., N, R_0 and C are set at the nominal values

and not changed during the simulation). It is shown that the settling time of the RED is longer than that of the PI controller, the AVQ and the MU_OPT controller. Fig. 6.9 also justifies the conclusion that the queue size oscillations of the RED and the PI controller are larger than that of the MU_OPT controller, which in turn is larger than that of the AVQ. But the AVQ does not have the capability to clamp the queue size to the reference value of 200 packets, whereas the PI controller and the MU_OPT controller can achieve this.

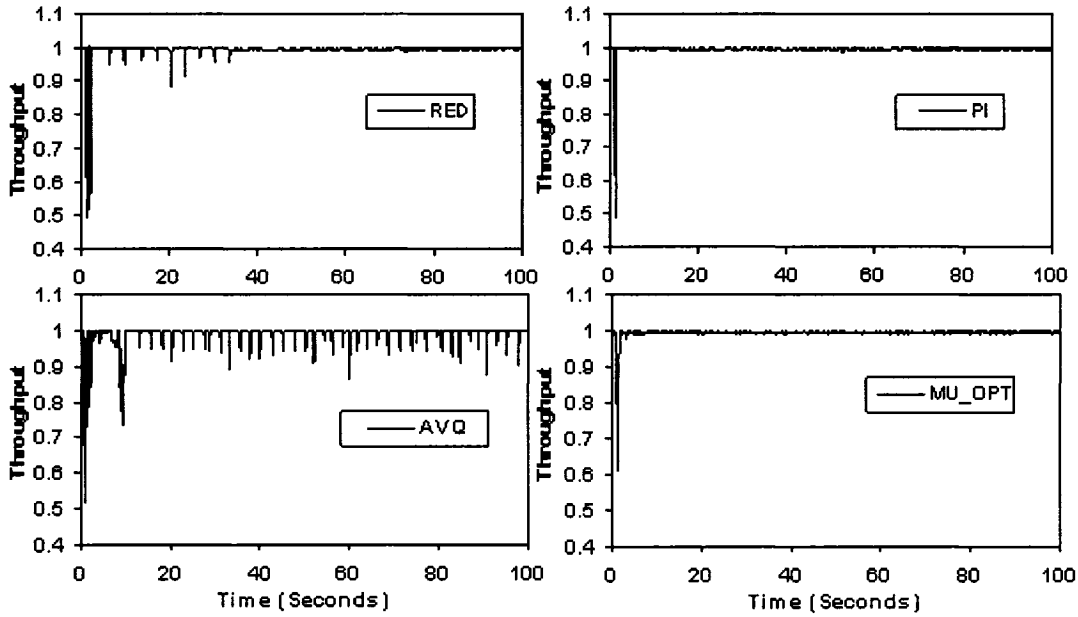

Fig. 6.10: Router throughput vs. Time: comparison of the RED, the PI, the AVQ and the MU_OPT controllers

Fig. 6.10 presents the bottleneck router throughput under the four controllers. The router throughput is defined as the number of packets forwarded by the router divided by the number of packets entering the router within a specific period of time. At the beginning of the simulation, the RED, the AVQ and the PI controller show a bottom value of 0.5, whereas the MU_OPT controller presents a relatively higher one (0.61). After the initial start-up period, the PI controller and the MU_OPT controller give similar throughput variations, within the range from 0.995 to 1. On the other hand, the RED controller shows a relatively larger range from 0.89 to 1, and the AVQ shows the largest range from 0.74 to 1.

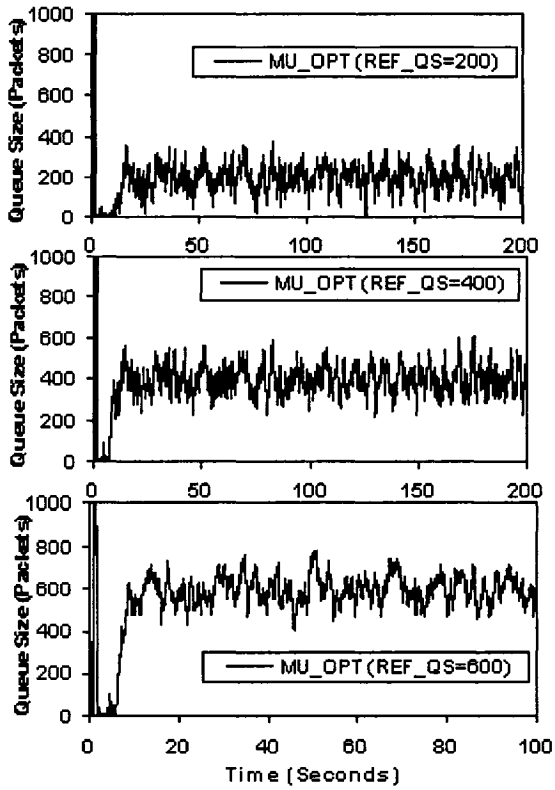

Fig. 6.11: Instantaneous queue size vs. Time: comparison of the MU_OPT controller with different reference queue sizes (REF_QS=200, 400, 600)

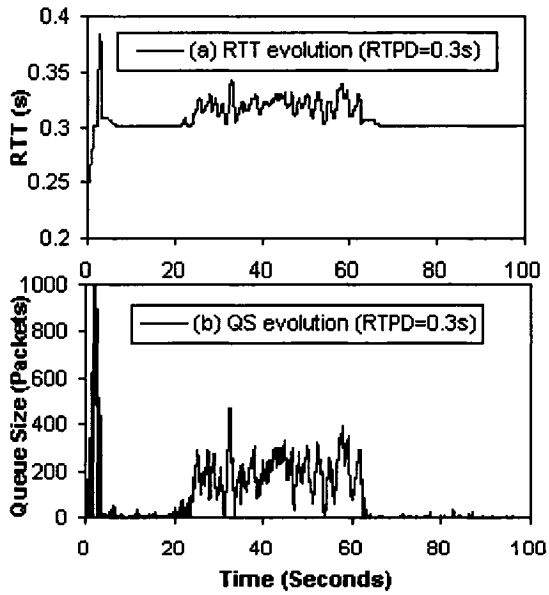

Fig. 6.12: RTT and instantaneous queue size evolutions when the RTPD is set to 0.3s

Experiment 2 – Different REF_QS

Now we investigate the case when the reference queue size (REF_QS) is set to different

values under the MU_OPT controller. Fig. 6.11 depicts the instantaneous queue size when REF_QS=200, 400, 600 packets. It is shown that the MU_OPT controller can successfully clamp the queue size at the reference value. It is well known that larger REF_QS means a longer average packet delay but a higher buffer utilization. To seek a tradeoff between delay and utilization, the recommended REF_QS is 20%~60% of the buffer size.

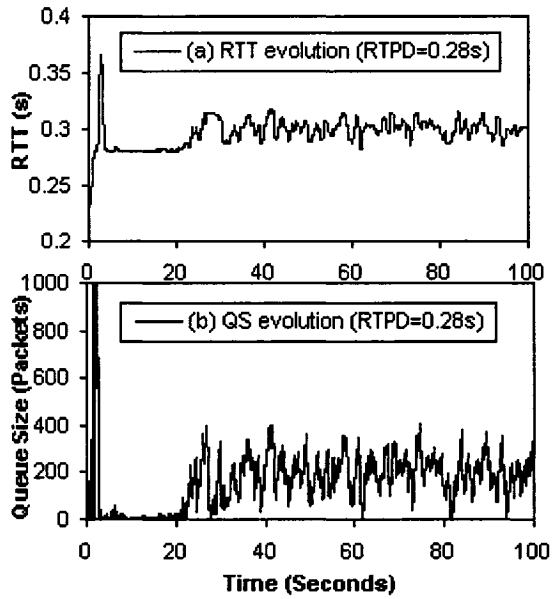

Fig. 6.13: RTT and Instantaneous queue size evolutions when RTPD=0.28s

Experiment 3 – Verification of the theoretical stability range for R_0; Time-varying RTT

In this experiment, we would like to investigate the MU_OPT controller performance when round trip time deviates from the nominal value. In Section 6.2.3, we have determined that the calculated theoretical stability range for R_0 is $R_0 \leq 0.315$s. Here we would verify this through simulations. We know that R_0 (Round Trip Time or RTT) is equal to RTPD (Round Trip Propagation Delay) plus the queuing delay. We have run lots of simulations with different values of RTPD. The following are the two critical cases: (i) RTPD=0.3s; (ii) RTPD=0.28s. Both N and C are fixed at their nominal values. Fig. 6.12 and Fig. 6.13 illustrate the RTT evolution and queue size evolution for case (i) and (ii), respectively. From Fig. 6.12 (a), during the period 24s~63s, the average RTT is equal to 0.32s, which is greater than 0.315s. From the theoretical calculation, the system should become unstable, and Fig. 6.12 (b) verifies this. Fig. 6.12 (b) shows that after 63 seconds, the queue size cannot be stabilized at the reference value of 200 packets. However, when RTPD=0.28s, the average RTT is equal to 0.3s after the initial start-up period from Fig. 6.13 (a), and it is within the

theoretical stability range. Fig. 6.13 (b) verifies that the system is stable because the queue size can be clamped around the reference value of 200 packets after the initial start-up period.

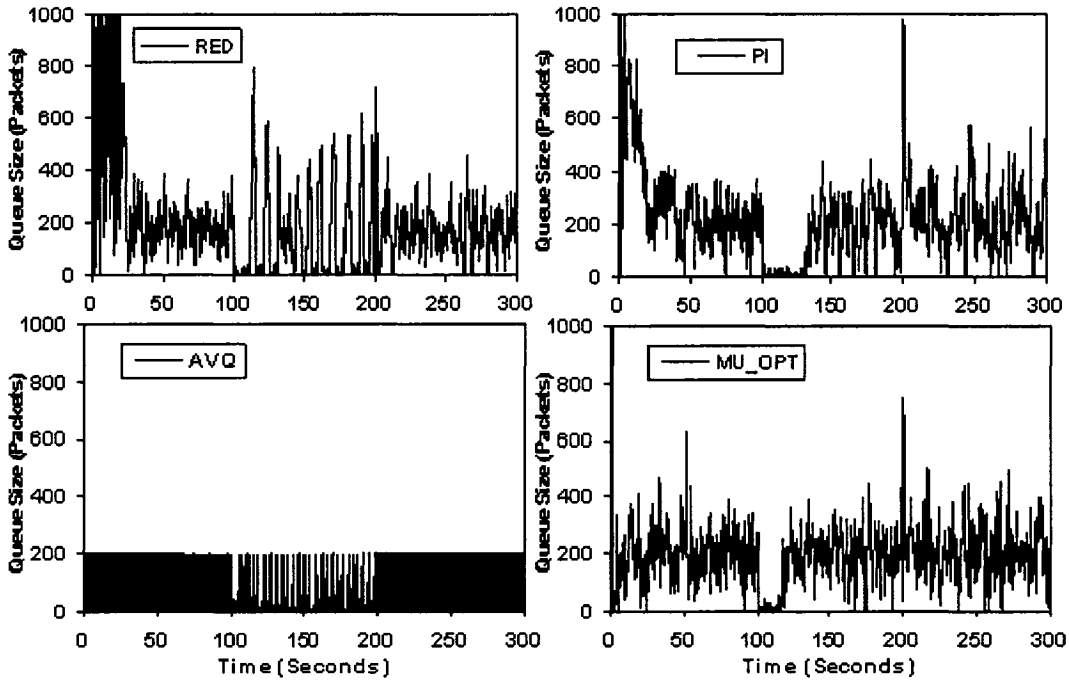

Fig. 6.14: Instantaneous queue size vs. Time: comparison of the RED, the PI, the AVQ and the MU_OPT controllers in the case of time-varying RTTs, RTPD varies from 100ms (0 - 100 sec), 280ms (100 - 200 sec) to 50ms (200 - 300 sec)

In order to investigate the MU_OPT controller performance when the RTT is time varying, we now change the RTPDs of all connections with the time: from 0 to 100 seconds, all the RTPDs are set as 100ms; at the time of 100 seconds, they are increased to 280ms and are kept at this value until the simulation time of 200 seconds, when all the RTPDs are decreased to 50ms. The instantaneous queue sizes under the RED, the PI controller and the MU_OPT controller are given in Fig. 6.14. In the first simulation period (0~100 sec), all the four controllers can stabilize the queue size, but the settling times are different: 30 seconds (RED), 25 seconds (PI), 0 second (AVQ), and 7 seconds (MU_OPT). Although settling time for the AVQ is short, it is at the cost of large packet loss rate, which is shown in Fig. 6.15. In the second simulation period (100~200 sec), the queue size under the RED shows larger oscillations than that under the PI controller, the AVQ and the MU_OPT controller. The settling time under the PI (35 sec) is larger than that under the MU_OPT (19sec), which in

turn is larger than that under the AVQ (8 seconds). When it comes to the third period (200~300sec), all four controllers can quickly stabilize the queue size, and the settling time under the PI controller is larger than that under the RED, the AVQ and the MU_OPT controller. But the AVQ shows large packet loss rate again in the third simulation period, and Fig. 6.15 confirms this. The AVQ does not have the capability to clamp the queue size to the REF_QS of 200 packets in all the three simulation periods.

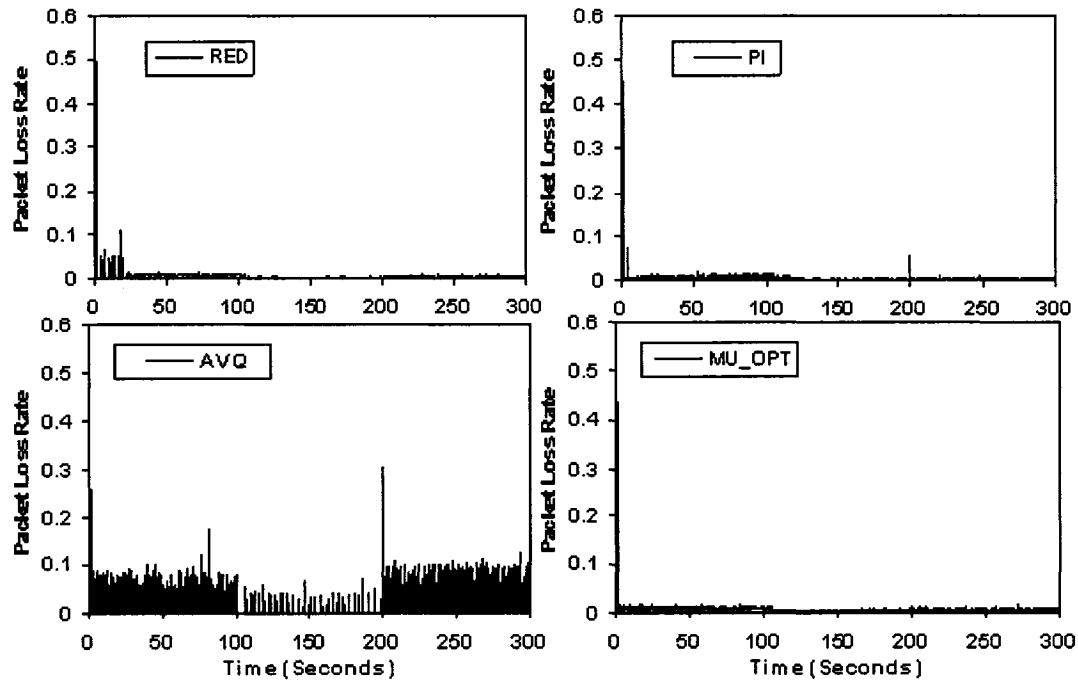

Fig. 6.15: Packet loss rate vs. Time: comparison of the RED, the PI, the AVQ and the MU_OPT controllers in the case of time-varying RTTs, RTPD varies from 100ms (0 - 100 sec), 280ms (100 - 200 sec) to 50ms (200 - 300 sec)

Fig. 6.15 illustrates the packet loss rate comparison. It shows that, after the initial start-up period, the RED, the PI controller and the MU_OPT controller have similar variations in packet loss rate: within the range from 0% to 1%, but the AVQ shows a significant larger range: most values are from 0% to 10% during the period of 0~100 sec and 200~300 sec.

Experiment 4 - Different RTTs for different connections

In this experiment, different connections have different RTTs. Totally we have 100 ftp connections and 20 http connections. The RTPDs are set as follows. For the ftp sources, the RTPDs are 100ms, 200ms, 300ms and 400ms for the connection 1~25, connection 26~50, connection 51~75 and connection 76~100 respectively, and for the http sources, the RTPDs

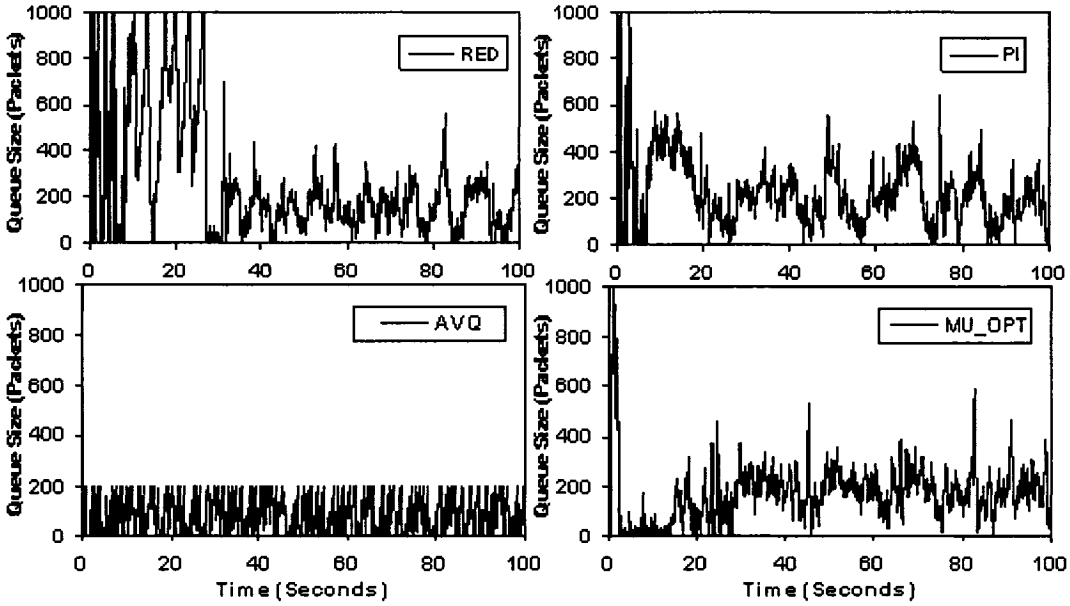

Fig. 6.16: Instantaneous queue size vs. Time: comparison of the RED, the PI, the AVQ and the MU_OPT controllers in the case of different RTTs for different connections

are set to 100ms and 400ms for connection 1~10 and connection 11~20 respectively. The instantaneous queue size is given in Fig. 6.16. It is shown that all the four controllers can stabilize the queue size, but with different settling times. The RED takes 30 seconds to stabilize the queues size, whereas the PI controller and the MU_OPT controller take 7 and 14 seconds, respectively, to clamp the queue size to the steady state. The AVQ takes almost zero second to clamp the queue size to a low value of 100 packets, which is far from the REF_QS of 200 packets. The queue size oscillations under the PI controller are larger than those under the MU_OPT controller and the AVQ.

Experiment 5 - Verification of the theoretical stability range for N; Varying N

In this experiment, we would like to investigate the MU_OPT controller performance when the number of long-lived connections (N) deviates from the nominal value.

The theoretical stability range for N when varying N alone has been determined to be $N \geq 71$ in Section 6.2.3. Here we would determine the stability range for N through simulations. We have run many simulations with different N while R_0 and C are kept at the nominal values. It has been found that the stability range for N is $N \geq 73$, which is very close to the theoretical stability range. Fig. 6.17 shows the queue size evolutions when N is set as 72 and 73. It is shown that when N=72, the controller cannot stabilize the queue size to the reference

value of 200 packets after 36 seconds. However when N=73, the queue size can be clamped around 200 packets after the initial start-up period.

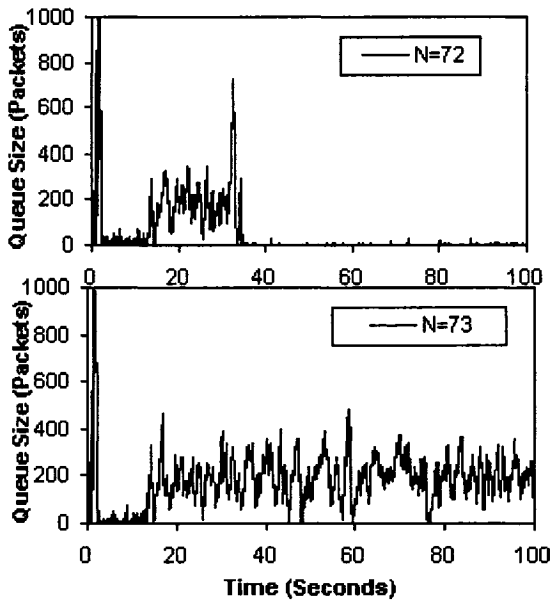

Fig. 6.17: Instantaneous queue size evolutions when N=72 and 73

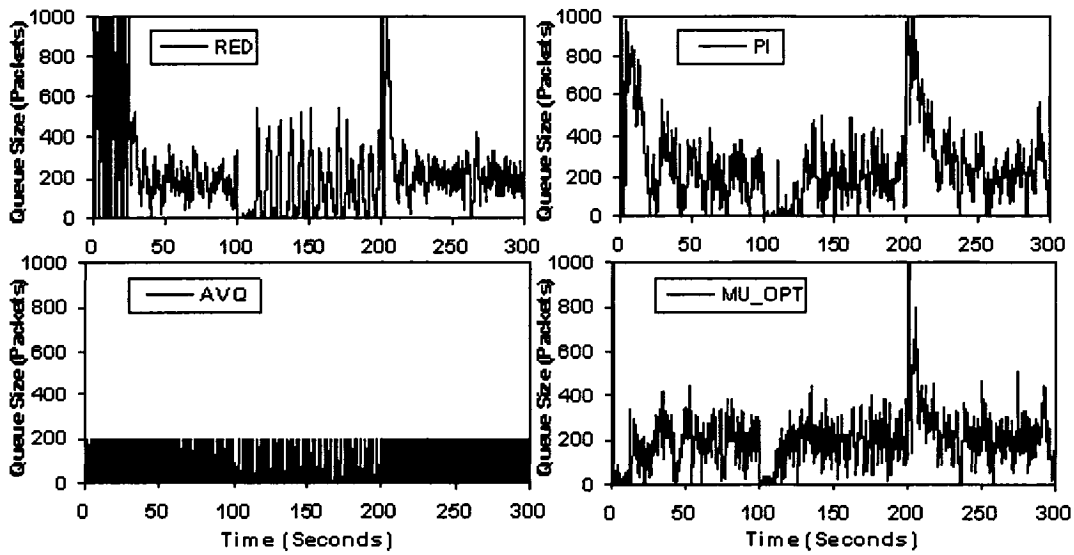

Fig. 6.18: Instantaneous queue size vs. Time: comparison of the RED, the PI, the AVQ and the MU_OPT controllers in the case of varying the number of long-lived connections (N)

We now change the number of long-lived connections (N) with time. From 0 to 100 seconds, we have 150 connections; at the time of 100 seconds, it is decreased to 80 and is kept at this value until 200 seconds, when the number of connections is increased to 200. The instantaneous queue size is shown in Fig. 6.18. In the first simulation period (0~100 sec), all

the four controllers can stabilize the queue size, but the settling times are different: 35 seconds (RED), 26 seconds (PI), 0 second (AVQ) and 13 seconds (MU_OPT). Although settling time for the AVQ is short, it is at the cost of large packet loss rate, which can be inferred from Fig. 6.19. In the second simulation period (100~200 sec), the queue size under the RED shows larger oscillations than that under the PI controller, the AVQ and the MU_OPT controller. The settling time under the PI controller (28 sec) is larger than that under the MU_OPT controller (13 sec), which in turn is larger than that under the AVQ (5 sec). When it comes to the third period (200~300sec), all four controllers can stabilize the queue size, and the PI controller and the MU_OPT controller shows similar settling time, which is larger than that under the RED. The AVQ shows the shortest settling time, but again it is at the cost of large packet loss rate (refer to Fig. 6.19). In addition, the AVQ does not have the capability to clamp the queue size to the REF_QS of 200 packets during the whole simulation time.

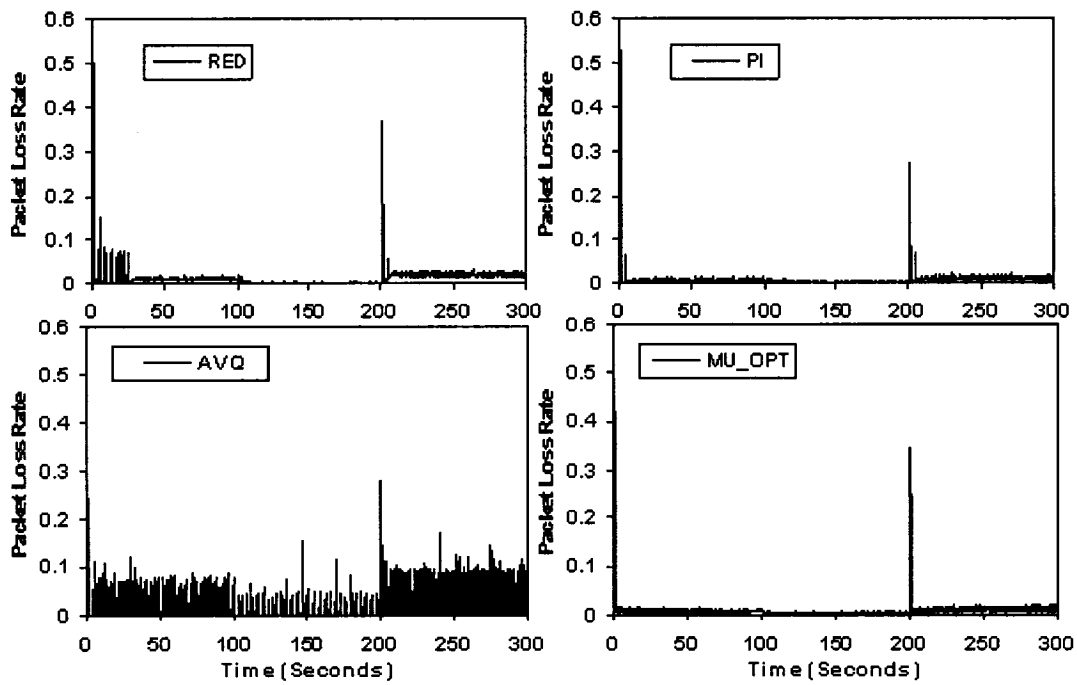

Fig. 6.19: Packet loss rate vs. Time: comparison of the RED, the PI, the AVQ and the MU_OPT controllers in the case of varying the number of long-lived connections (N)

The packet loss rate comparison is provided in Fig. 6.19, which shows that, after the initial start-up period, the RED, the PI controller and the MU_OPT controller give similar variations in packet loss rate: within the range from 0% to 1% during the first and the second

simulation period, and 0% to 3% during the third simulation period, but the AVQ shows a significant larger range: most packet loss rate values are from 0% to 10% during the simulation period of 0~100 sec and 200~300 sec.

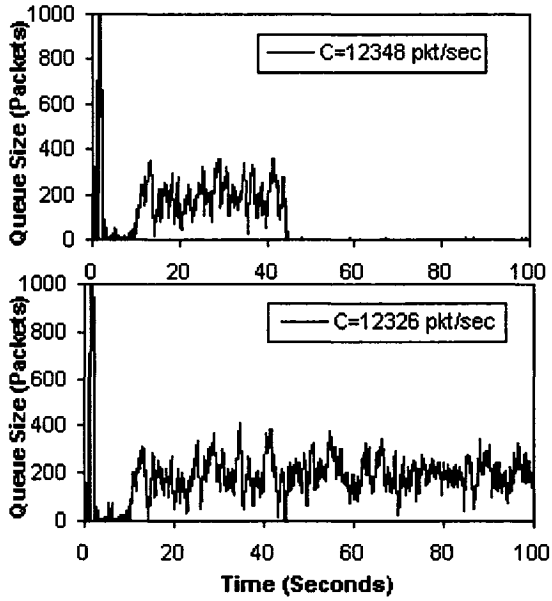

Fig. 6.20: Instantaneous queue size evolutions when C=12348pkt/s and 12326pkt/s

Experiment 6 - Verification of the theoretical stability range for *C*; Varying *C*

Now we change the link capacity C to values other than the nominal value. Section 6.2.3 determines that the theoretical stability range for C when varying C alone is $C \leq 12233$ packet/sec. To determine the simulation stability range for C, we have run many simulations with different C while N and R_0 are kept at their nominal values. The simulation stability range for C is then found to be $C \leq 12326$ packet/sec, which is close to the theoretical stability range.

Fig. 6.20 gives the queue size evolutions when C is equal to 12348 packet/sec and 12326 packet/sec. It shows that when C=12348 packet/sec, the queue size cannot be stabilized to the reference value of 200 packets after 46 seconds. However when C=12326 packet/sec, the queue size can be clamped around 200 packets after the initial start-up period.

We then vary the link capacity (C) with time. The link capacity is set as: 45Mbps (0-50s), 15Mbps (50s-100s) and 55Mbps (100s–150s). Fig. 6.21 illustrates the instantaneous queue size. In the first simulation period (0~50s), it takes the RED 30 seconds to stabilize the queue size, whereas the PI controller takes 20 seconds, the AVQ takes 0 second, and the MU_OPT controller takes 15 seconds. Although the settling time of the AVQ is short, it

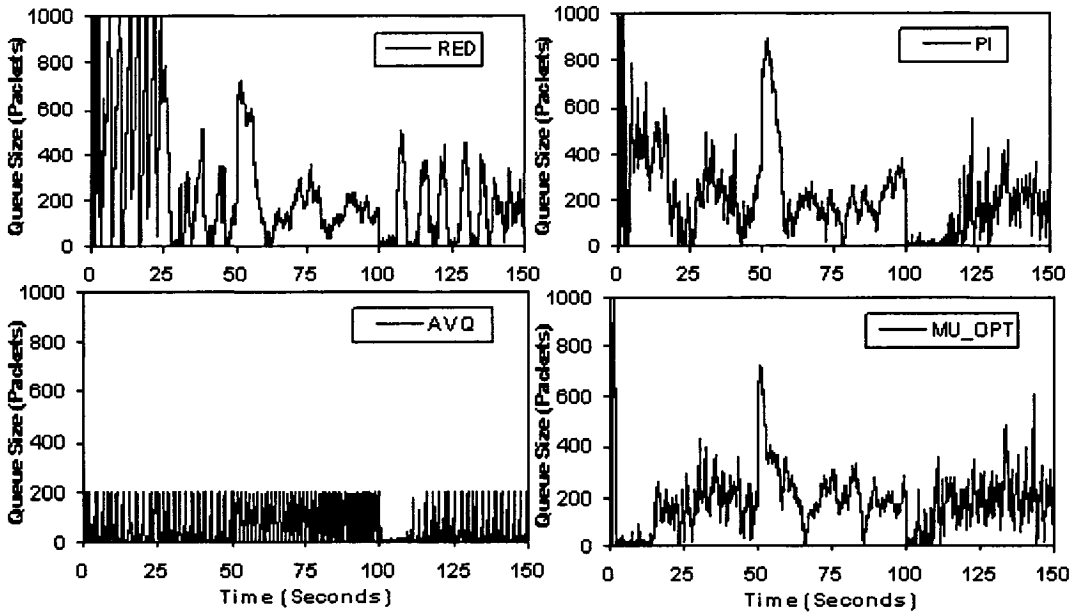

Fig. 6.21: Instantaneous queue size vs. Time: comparison of the RED, the PI, the AVQ and the MU_OPT controllers in the case of varying the link capacity (C)

cannot clamp the queue size to the REF_QS of 200 packets. In the second (50~100s) simulation period, all the four controllers can stabilize the queue size and the three controllers (the RED, the PI controller and the MU_OPT controller) shows similar settling time, which is larger than that of the AVQ. In the third (100~150s) simulation period, the queue size under the RED shows larger oscillations than that under the PI controller, the AVQ and the MU_OPT controller. The settling time under the PI controller (21s) is larger than that under the AVQ controller (12s) and the MU_OPT controller (11s).

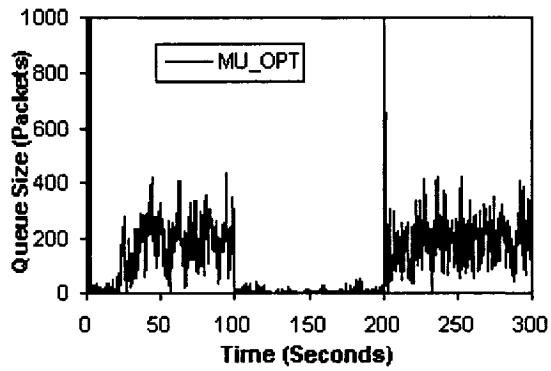

Fig. 6.22: Instantaneous queue size evolutions in the case of time-varying N, R_0 and C

Experiment 7 – Varying N, R_0 and C simultaneously

In this experiment, we vary N, R_0 and C at the same time. In Section 6.2.3, we have

theoretically determined the stability under the 3 combinations of N, R_0 and C: (a) stable: N=150, R_0=0.32s, C=8500 packet/sec; (b) unstable: N=80, R_0=0.3s, C=11000 packet/sec; (c) stable: N=100, R_0=0.12s, C=10000 packet/sec. We would verify this through simulation. For the three simulation periods 0~100s, 100~200s and 200~300s, we set the configuration as in (a), (b) and (c), respectively.

Fig. 6.22 presents the queue size evolution. It is shown that in the first simulation period 0~100s (configuration (a)), after the initial start-up period, the controller can stabilize the queue size around the reference value of 200 packets. However, in the second simulation period 100~200s when N, R_0 and C are changed to the values in configuration (b), the queue size cannot be stabilized to the reference value any more. When it comes to the third simulation period 200~300s (configuration (c)), the queue size can be clamped around 200 packets again.

6.4 Concluding Remarks

This chapter exploits the modern robust μ analysis and $\mathcal{H}_\infty$ optimal control theory to propose the MU_OPT controller design process for AQM routers. Considering explicitly the uncertainties in the TCP/AQM system, we have determined the uncertainty model using a graphical technique. The μ–optimal control problem is approximated by the $\mathcal{H}_\infty$ $S/T/U$ mixed sensitivity problem (MSP), which can be easily solved using a state-space solution. In addition, we have quantitatively analyzed the NP, the RS and the RP of the system under the synthesized controller using the μ analysis technique and verified the analysis through simulations. Furthermore, the effectiveness of the controller has been verified through extensive simulations including the case of varying N alone, R_0 alone, C alone or varying them together.

Chapter 7 Design Guideline

Based on our experience, we would like to propose a design guideline for the AQM methods in computer communication networks. This is first presented in the following Table 7.1 on the comparison of different controllers proposed in this thesis in terms of the self-tunability / real-time capability, the design complexity, the ability to clamp the average queue size to a reference value, the ability to bound the closed-loop dynamics explicitly, the ability to analyze NP, RS and RP quantitatively, their design parameter operation ranges, and the network scenario each controller may apply to.

Table 7.1 Comparison of different controllers proposed in this thesis

Controller	P_PP	PI_PP	ST_P_PP	ST_PI_PP	*S/U* MSP	MU_OPT
Self-tunability / Real-time capability	No	No	Yes	Yes	No	No
Ability to clamp the avg. queue size to a reference value?	No	Yes	No	Yes	Yes	Yes
Bound the closed-loop dynamics explicitly?	No	No	No	No	Yes	Yes
Ability to analyze NP, RS and RP quantitatively?	No	No	No	No	No	Yes
Ability to determine stability ranges for N, R_0, and C theoretically?	No	No	No	No	No	Yes
Design complexity	low	Medium	low	medium	high	high
Design parameter operation range	$0.2 \le \xi \le 0.8$	$0.2 \le \xi \le 0.8$; $1.0 \le \omega_n \le 2.6$	$\xi_0=0.5$; $\xi \in (0.2, 0.8)$	$\xi_0=0.5$; $\xi \in (0.2, 0.8)$; $\omega_{n0}=1.8$; $\omega_n \in (1.0, 2.6)$	W_U is in the order of $1/p_0$ plus 2 or 3; $\omega_B \in (2.5,5)$; $M \in (1.5,2)$; A is in the order of -4 or -5	W_M is determined by uncertainty modeling; W_U, ω_B, M and A are the same as in *S/U* MSP
Network scenario the controller may apply to	Simple network with very small variations on the network parameters	Scenario in P_PP + requirement to clamp the steady-state average queue size to a reference value	Network parameters exhibit large oscillations	Scenario in ST_P_PP + requirement to clamp the steady-state average queue size to a reference value	Variations of the network parameters are medium	Scenario in *S/U* MSP + range of uncertainties of the network parameters can be estimated

In summary, we propose the following design guideline:

1) For a simple network with very small variations on the network parameters (N, R_0 and C), we may use the P_PP controller because it is simple in the sense that there is just one control parameter K_P and the user is required to specify only one performance index, namely the damped ratio ξ. If we want to clamp the steady-state average queue size to a reference value so that the delay experienced by the packets has small variations and the high link utilization can be achieved, we could adopt the PI_PP controller because this controller has good track performance. In this case, the user has to specify two performance indices ξ and ω_n, in order to determine the two controller parameters K_P and K_I.
2) If the network parameters exhibit large oscillations, the self-tuning controllers – the ST_P_PP controller or the ST_PI_PP controller are recommended. For the ST_P_PP controller, the user just needs to specify the initial value of the damped ratio ξ, and the controller will keep on-line tuning of only one controller parameter K_P. However, an offset exists between the average queue size and the reference value. Using the ST_PI_PP controller can eliminate the offset, and the user needs to specify the initial values of the two performance indices ξ and ω_n. In addition, two controller parameters K_P and K_I are monitored and on-line tuned by the ST_PI_PP controller.
3) If the variations of the network parameters are medium, we recommend the usage of the *S/U* MSP controller. Because the closed-loop dynamics of the system can be explicitly bounded by the weight functions in the *S/U* MSP controller, and no on-line tuning burden is incurred on the router.
4) If we can estimate the range of uncertainties of the network parameters, the MU_OPT controller will be a good choice. The uncertainty modeling in the design of the MU_OPT controller requires the knowledge of the range of uncertainties of the network parameters. If the actual variations of the network parameters are indeed within the estimated ranges, then the MU_OPT controller can ensure the robust stability of the system. In fact, the MU_OPT controller is the only one (among the others in this thesis) that can determine the stability ranges for the network parameters, e.g., N, R_0 and C theoretically, and the determination method can be found in Section 6.2.3, where the relationship between the parameter uncertainties and the system stability has been analyzed.

5) Self-tunable MU_OPT controller and *S/U* MSP controller are not recommended at this writing because we need to use the MATLAB software to synthesize the controller. In practice, it is unlikely that we install the whole MATLAB software on a router to solve this type of modern robust AQM controllers on-line. Of course, this would become feasible if we could install a light-weight version of MATLAB software that only has the toolbox to synthesize the modern robust controller, or if we can use the C language to implement the toolbox by ourselves.

Chapter 8 Conclusion

As set out in our objectives at the beginning of this thesis, we have designed several AQM algorithms for IP routers supporting TCP flows based on the control theoretical approaches. We have presented and justified the design formulae and the design procedures. Extensive OPNET simulations have been used to verify the effectiveness of our designed controllers, and to provide a design guideline.

We have first designed the controllers without self-tuning feature – the P_PP controller and the PI_PP controller, based on the pole placement technique of the classical control theory. We then have implemented the self-tuning controllers – the ST_P_PP controller and the ST_PI_PP controller. These controllers have the capability to tune the controller parameters K_P and K_I on-line when ξ or ω_n deviates from the preset intervals, and thus to ensure the stability and satisfactory performance of the system even under significant load changes and RTT changes.

Based on the modern $\mathcal{H}_\infty$ optimal control theory, we have designed the *S/U* MSP controller and the *S/T/U* MSP controller. Under these controllers, with appropriately chosen weight functions, the closed-loop dynamics of the TCP/AQM system can be shaped as desired, and thus the satisfactory system performance is achieved. Through theoretical analysis and simulations, we have formulated and justified the guidelines for the choices of the weight functions W_S, W_U and W_T.

Finally we have devised the MU_OPT controller based on the robust μ–analysis technique. Through a graphical technique, the uncertainties of the TCP/AQM system were explicitly modeled by the uncertainty weight. We have also tackled the μ–optimal control problem with the $\mathcal{H}_\infty$ *S/T/U* mixed sensitivity problem, and solved it with a state-space solution. In addition, the NP, the RS and RP of the system were analytically evaluated through the μ analysis technique, and were verified through OPNET simulations.

As a closing statement, it might be valuable to make a comparison between the classical control methods and the modern control methods. In general the modern control methods require a higher design complexity than the classical ones. Therefore it is very difficult for the modern control methods to produce self-tunable controllers. However, the

modern control methods also show some advantages over the classical ones. For example, we can bound the closed-loop dynamics explicitly (the *S/U* MSP controller, the MU_OPT controller) and analyze the NP, RS and RP quantitatively (the MU_OPT controller) using the modern control methods, whereas these cannot be achieved when the classical methods are used.

There have been various difficulties and lessons learned during the course of this work.

1) *How to derive the mathematic design formulae for the P_PP controller and the PI_PP controller?* When we first started our work on this research topic, we found it difficult to come up with a set of formulae that could relate the transient performance indices ξ, ω_n directly with the controller parameter K_P, K_I. It turned out that Assumption (3.2) can be used to analyze the problem into three cases according to the value of ξ. The PI_PP controller was much more difficult to solve. So after investigating many methods, we made the observation (3.29) and achieved a very fine and elegant solution for the PI_PP controller.
2) *How to formulate the $\mathcal{H}_\infty$ optimal control problem for the TCP/AQM system and how to solve it?* Although the $\mathcal{H}_\infty$ optimal control appears to be an interesting approach to us, we have to study many difficult formulations in order to solve the AQM controllers for the TCP/AQM system. After many trials and errors, we finally formulated it using the *S/U* MSP and *S/T/U* MSP, and solved it using the state-space solution. In the MSP design for the AQM controller, the key part is to choose appropriate weights, especially the input sensitivity weight W_U. After careful reasoning and deliberation, we finally came up with the wise insights on how to choose W_U, which was described in Section 5.2.
3) *How to determine the theoretical stable ranges in the MU_OPT controller design?* In the MU_OPT controller design, when we were trying to verify the RS (Robust Stability) of the system, we need to determine the theoretical stable ranges, which is not a straightforward issue. It took us much effort to obtain the method described in Section 6.2.3 to determine the theoretical stable ranges.

Although we came a long way from trial and error experimenting with different options, we are happy to come up with some design guidelines for future endeavors.

8.1 Future Work

For future work, we can extend our study in this thesis to address the following items:

1) A practical protocol that can describe and implement all proposed algorithms according to the industrial standards.
2) Apply the control approaches utilized in this thesis to rate-based control schemes, e.g., to the TCP-friendly congestion control schemes.
3) A PID type of controller that can achieve both good dynamic performance and good steady performance.
4) Evaluate the proposed algorithms more thoroughly under more scenarios. For example, investigate the system performance under the multi-bottleneck link configuration.

References

[AlPa99] M. Allman, V. Paxson and W. Stevens, "TCP congestion control," RFC 2581, IETF, Apr. 1999.

[AsHa95] K. J. Astrom and T. Hagglund, *PID Controllers: Theory, Design, and Tuning*, Instrument Society of America, Research Triangle Park, NC, 1995.

[AsWi84] Karl J. Astrom and Bjom Wittemnark, *Computer Controlled Systems: Theory and Design*, Prentice-Hall, 1984.

[AwOu01] J. Aweya, M. Ouellette, and D. Y. Montuno, "A control theoretic approach to active queue management," *Computer Networks*, Vol. 36, No. 2–3, Jul. 2001, pp. 203–35.

[BeZh96] J. C. R. Bennett and H. Zhang, "WF2Q: worst-case fair weighted fair queuing," *Proceedings of IEEE INFOCOM*, San Francisco, CA, Mar. 1996, pp. 120-128.

[Bode40] H.W. Bode, "Feedback Amplifier Design," *Bell System Tech. J.*, Vol. 19, 1940, pp. 42.

[BoFe95] F. Bonomi and K. Fendick, "The rate-based flow framework for the available bit rate ATM service," IEEE Network Magazine, Vol. 9, 1995, pp. 25-39.

[BoMa00] T. Bonald, M. May, and J. C. Bolot, "Analytic evaluation of RED performance," *Proceedings of IEEE/INFOCOM*, 2000, pp. 1415–1424.

[Brad98] B. Braden et al. "Recommendations on queue management and congestion avoidance in the Internet," *RFC2309*, IETF, Apr. 1998.

[BrMa94] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson, "TCP vegas: new techniques for congestion detection and avoidance," *Proceedings of ACM SIGCOMM*, London, U.K., Aug. 1994, pp. 34-35.

[ChCl95] A. Charny, D. Clark, and R. Jain, "Congestion control with explicit rate indication," *Proceedings of ICC'95*, Jun. 1995, pp. 1954-1963.

[ChJe00] M. Christiansen, K. Jeffay, D. Ott, and F. D. Smith, "Tuning RED for web traffic," *Proceedings of the ACM SIGCOMM*, 2000, pp. 139–150.

[Cisc98] Technical Specification from Cisco, *Distributed Weighted Random Early Detection*, URL: http://www.cisco.com/univercd/cc/td/doc/product/software/ios111/cc111/wred.pdf.

[Clar88] D.D. Clark, "The design philosophy of the DARPA Internet Protocols," *Proceedings of ACM SIGCOMM*, Stanford, CA, Aug. 1988, pp.106-114.

[ClFa98] D. Clark and W. Fang, "Explicit allocation of best effort packet delivery service, " *IEEE/ACM Transactions on Networking*, Vol. 6, No. 4, Aug. 1998, pp. 362 – 373.

[ClHo95] D.D. Clark and J. Hoe. "Start-up dynamics of TCP's congestion control and avoidance schemes," *Technical report*, Presentation to the Internet End-to-End Research Group, Jun. 1995.

[DeKe89] A. Demers, S. Keshav, and S. Shenker, "Analysis and simulation of a fair queuing algorithm," *Proceedings of the ACM SIGCOMM*, Austin, TX, Sept. 1989, pp. 1-12.

[DoGl89] J.C. Doyle, K. Glover, P.P. Khargonekar, and B.A. Francis, "State-Space Solutions to Standard $\mathcal{H}_2$ and $\mathcal{H}_\infty$ Control Problems, " *IEEE Transactions on Automatic Control*, Vol. 34, No. 8, Aug. 1989, pp.831-847.

[DoSt81] J.C. Doyle, and G. Stein, "Multivariable Feedback Design: Concepts for a Classical/Modern Synthesis," *IEEE Transactions on Automatic Control*, Vol. AC-26, Feb. 1981, pp. 4-16.

[Doyl82] J. C. Doyle, "Analysis of feedback systems with structured uncertainties," *IEE Proceedings, Part D* 129(6), 1982, pp. 242-250.

[Doyl83] J.C. Doyle, "Synthesis of Robust Controllers and Filters, " Proceeding *of IEEE Conference on Decision and Control*, San Antonio, Texas, 1983, pp. 109-114.

[Evan48] W.R. Evans, "Graphical Analysis of Control Systems," *Transactions AIEE*, Vol. 67, 1948, pp. 547-551.

[FaFl96] K. Fall and S. Floyd, "Simulation-based comparisons of Tahoe, Reno and SACK TCP," *ACM SIGCOMM Computer Communication Review*, Vol. 26, No. 3, Jul. 1996, pp. 5-21.

[FeKa99] W.C. Feng, D.D. Kandlur, D. Saha, and K.G. Shin, "A self-configuring RED gateway," *Proceedings of IEEE/INFOCOM*, 1999, pp. 1320–1328.

[FeSh02] W.C. Feng, K.G. Shin, D.D. Kandlur, and D. Saha, "The blue active queue management algorithms," *IEEE Transaction on Networking*, Vol. 10, No. 4, Aug. 2002, pp. 513–528.

[FiBo00] V. Firoiu and M. Borden, "A study of active queue management for congestion control, " *Proceedings of IEEE/INFOCOM,* 2000, pp. 1435 -1444.

[FlFa97] S. Floyd and K. Fall, "Router mechanisms to support end-to-end congestion control.", *Tech. rep, LBL*, 1997. URL: http://wwwnrg.ee.lbl.gov/nrg-papers.html.

[FlFa99] S. Floyd and K Fall, "Promoting the use of end-to-end congestion control in the Internet," IEEE/ACM *Transaction on Networking*, Vol. 7, No. 4, Aug. 1999, pp. 458–472.

[FlHe04] S. Floyd, T. Henderson and A. Gurtov, "The NewReno Modification to TCP's Fast Recovery Algorithm," *RFC 3782, IETF*, Apr. 2004.

[FlJa91] S. Floyd and V. Jacobson, "Traffic phase effects in packet-switched gateways," ACM Computer Communication Review, Vol. 21, No. 2, Apr. 1991, pp. 26-42.

[FlJa93] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance", *IEEE/ACM Transactions on Networking*, Vol. 1, No. 4, Aug. 1993, pp. 397-413.

[Floy96a] S. Floyd. "Issues of TCP with SACK," *Technical report*, Mar. 1996. URL: ftp://ftp.ee.lbl.gov/papers/issues_sa.ps.Z.

[Floy96b] S. Floyd. "SACK TCP: The sender's congestion control algorithms for the implementation "sackl" in LBNL's "ns" simulator (viewgraphs)," *Technical report*, Presentation to the TCP Large Windows Working Group of the IETF, Mar. 1996. URL: ftp://ftp.ee.lbl.gov/talks/sacks.ps.

[Fu70] K.S. Fu. "Learning control systems – review and outlook, " *IEEE Transactions on Automatic Control*, Vol. 15, No. 2, Apr. 1970, pp. 210-221.

[GeCr01] P. Gevros, J. Crowcroft, P. Kirstein, and S. Bhatti, "Congestion control mechanisms and the best effort service model," *IEEE Network*, Vol. 15, No. 3, May/Jun. 2001, pp.16-26.

[Gole94] S.J. Golestani, "A self-clocked fair queuing scheme for broadband applications," *Proceedings of IEEE INFOCOM*, Toronto, Canada, June 1994.

[GoRa80] G. Goodwin, P. Ramadge, P. Caines, "Discrete-time multivariable adaptive control," *IEEE Transactions on Automatic Control*, Vol. 25, No. 3, Jun. 1980, pp. 449-456.

[HaFl03] M. Handley, S. Floyd, J. Padhye and J. Widmer, "TCP friendly rate control (TFRC): protocol specification," *RFC 3448, IETF*, Jan. 2003.

[Hash89] E. Hashem, "Analysis of random drop for gateway congestion control," *Technical Report MIT/LCS/TR-465*, Laboratory for Computer Science, MIT, Cambridge, MA, 1989.

[Hoe95] J. Hoe. "Start-up dynamics of TCP's congestion control and avoidance schemes," Master's thesis, MIT. Jun. 1995.

[HoMi01a] C.V. Hollot, V. Misra, D. Towsley and W.B. Gong, "A Control Theoretic Analysis of RED, " *Proceedings of IEEE/INFOCOM,* 2001, pp. 1510 -1519.

[HoMi01b] C.V. Hollot, V. Misra, D. Towsley and W.B. Gong, "On Designing Improved Controllers for AQM Routers Supporting TCP Flows, " *Proceedings of IEEE/INFOCOM,* 2001, pp. 1726 -1734.

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

[Jaco73] D. Jacobson, "Optimal stochastic linear systems with exponential performance criteria and their relation to deterministic differential games," *IEEE Transactions on Automatic Control*, Vol. 18, No. 2, Apr. 1973, pp.124-131.

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

[Jaco90] V. Jacobson, "Modified TCP congestion avoidance algorithm," *Technical report,* Email to the end2end-interest Mailing List, Apr. 1990., URL: ftp://ftp.ee.lbl.gov/email/vanj.90apr30.txt.

[KaBe60] R.E. Kalman and J.E. Bertram, "Control System Analysis and Design via the 'Second Method' of Lyapunov. I. Continuous-time Systems," *Transaction ASME Journal Basic Engineering*, Jun. 1960, pp. 371-393.

[Kalm60] R.E. Kalman, "Contributions to the theory of optimal control," *Bol. Soc. Mat. Mexicana*, Vol. 5, 1960, pp. 102-119.

[Kesh91] S. Keshav, "A control-theoretic approach to flow control, " *Proceedings of ACM SIGCOMM*, Zurich, Switzerland, Sep. 1991, pp. 3-15.

[KuSr04] S.S. Kunniyur and R. Srikant, "An adaptive virtual queue (AVQ) algorithm for active queue management," *IEEE Transaction on Networking*, Vol. 12, No. 2, Apr. 2004, pp. 286–299.

[Kwak93] H. Kwakernaak, "Robust Control and $\mathcal{H}_\infty$–optimization —Tutorial Paper," *Automatica*, Vol. 29, No. 2, 1993, pp.255-273.

[LaNe96] T.V. Lakshman, A. Neidhardt, and T.J. Ott, "The drop from front strategy in TCP and in TCP over ATM," Proceeding *of IEEE INFOCOM*, San Francisco, CA, Mar. 1996.

[LiMo97] D. Lin and R. Morris, "Dynamics of random early detection," *Proceedings of the ACM SIGCOMM*, 1997, 127–137.

[LoZh05] C. Long, B. Zhao, X. Guan, J. Yang, "The Yellow active queue management algorithm, " *Computer Networks*, Vol. 47, No. 4, Mar. 2005, pp. 525-550.

[Macc45] L.A. MacColl, *Fundamental Theory of Servomechanisms*, New York: Van Nostrand, 1945.

[MaMa96a] M. Mathis, J. Mahdavi, S. Floyd and A. Romanow, "TCP selective acknowledgement options," *RFC 2018, IETF*, Oct. 1996

[MaMa96b] M. Mathis and J. Mahdavi, "Forward acknowledgement: refining TCP congestion control," *SIGCOMM Symposium on Communications Architectures and Protocols*, Aug. 1996.

[MaSe97] M. Mathis, J. Semke, J. Mahdavi, and T. Ott, "The macroscopic behavior of the TCP congestion avoid algorithm," *ACM Computer Communication Review*, Vol. 27, No. 3, Jul. 1997.

[MATL02] MATLAB Products, URL: http://www.mathworks.com.

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

[Neid96] A.L. Neidhardt, "Traffic Source Models for the Bestavros and Crovella data," *Private Communication*, 1996.

[Ogat97] K. Ogata, *Modern Control Engineering*, 3rd Edition, Prentice Hall, 1997.

[OhMu95] H. Ohsaki, M. Murata, and et al, "Rate-based congestion control for ATM Networks," *ACM Computer Communication Review*, Vol. 25, No. 2, 1995, pp. 60-72.

[OPNE01] OPNET University Program, URL: http://www.opnet.com/services/university/.

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

[PaGa93] A.K. Parekh and R.G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the single node case," *IEEE Network*, Vol. 1, No. 3, Jun. 1993, pp. 344-357.

[PaGa94] A.K. Parekh and R.G. Gallager, "A generalized processor sharing approach to flow control in integrated services networks: the multiple node case," *IEEE Network*, Vol. 2, No. 2, Apr. 1994, pp. 137-150.

[QuOz04] P.F. Quet and H. Özbay, "On the Design of AQM Supporting TCP Flows Using Robust Control Theory", *IEEE Transactions on Automatic Control*, Vol. 49, No. 6, Jun. 2004, pp.1031-1036.

[RaFl99] K. Ramakrishnan and S. Floyd, "A proposal to add Explicit Congestion Notification (ECN) to IP," RFC 2481, IETF, Jan. 1999.

[SaLa81] M.G. Safonov, A.J. Laub, and G.L. Hartmann, "Feedback Properties of Multivariable Systems: The Role and Use of the Return Difference Matrix," *IEEE Transaction on Automatic Control*, Vol. 26, No. 1, 1981, pp. 47-65.

[ShVa95] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round robin," *ACM Computer Communication Review*, Vol. 25, No. 4, Oct. 1995, pp. 365-386.

[SkPo01] S. Skogestad and I. Postlethwaite, *Multivariable Feedback Control, Analysis and Design*, John Wiley & Sons, 2001.

[Stev94] W.R. Stevens, TCP/IP Illustrated Volume I: The Protocols, Addison-Wesley 1994.

[Stev97] W.R. Stevens, "TCP slow start, congestion avoidance, fast retransmit, and fast recovery algorithms," *IETF, RFC 2001,* Jan. 1997

[SuLa98] B. Suter, T.V. Lakshman, D. Stiliadis, and A. Choudhury, "Efficient active queue management for Internet routers," *Proceeding of Engineering Conference at Interop 98*, Las Vegas, NV, May 1998.

[Tane88] A.S. Tanenbaum, *Computer Networks*, Prentice Hall, 1988.

[ToOz95] O. Toker and H. Özbay, "$\mathcal{H}_\infty$ optimal and suboptimal controllers for infinite dimensional SISO plants," *IEEE Transactions on Automatic Control*, Vol. 40, No. 4, Apr. 1995, pp.751-755.

[Tse71] E. Tse, "On the optimal control of stochastic linear systems," *IEEE Transactions on Automatic Control*, Vol. 16, No. 6, Dec. 1971, pp.776-785.

[WiDe01] J. Widmer, R. Denda, and M. Mauve, "A survey on TCP-friendly congestion control," *IEEE Network*, Vol. 15, No. 3, May/Jun. 2001, pp. 28-37.

[Wong92] S.S.M. Wong, *Computational Methods in Physics and Engineering*, Prentice Hall, 1992.

[Zame79] G. Zames, "Feedback and Optimal Sensitivity: Model Reference Transformation, Weighted Seminorms, and Approximate Inverses, " *Proceedings of 17th Allerton Conference*, 1979, pp.744-752.

[Zhan89] L. Zhang, "A new architecture for packet switching network protocols," *Report MIT/LCS/TR-455*, Laboratory for Computer Science, MIT, Cambridge, MA, Aug. 1989.

[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.

[ZhYa00] H. Zhang, O. Yang and H. Mouftah, "A hop-by-hop flow controller for a virtual path," *Computer Networks and ISDN Systems*, Vol. 32, 2000, pp. 99-119.

[ZhYa01] H. Zhang, O. Yang and H. Mouftah, "The distributed robust traffic controller for ATM networks," *International Journal of Communication System*, Vol. 14, 2001.

[ZhYa02] C. Zhu, O.W.W. Yang, and et al, "A comparison of active queue management algorithms using the OPNET Modeler," *IEEE Communications Magazine*, Vol. 40, No. 6, Jun. 2002, pp. 158–167.

[ZhYa97] H. Zhang, O. Yang and H. Mouftah, "Design of robust congestion controllers for ATM networks," *Proceedings IEEE INFOCOM*, 1997, pp. 302-309.

Appendix A Background on Pole Placement

We shall summarize the feedback control theory here to introduce the terminology used in Chapter 3 and Chapter 4 of the thesis. Details can be found in any standard control textbook like [Ogat97].

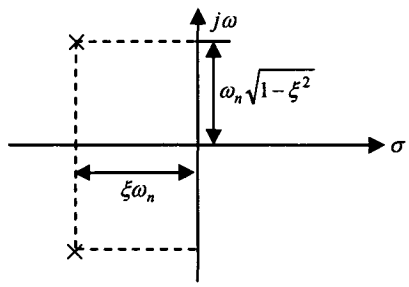

Fig. A.1: Complex conjugate poles when $0<\xi<1$

Suppose a closed-loop transfer function is

$$F(s)=\frac{\omega_n^2}{s^2+2\xi\omega_n s+\omega_n^2}\ , \tag{A.1}$$

where ξ, ω_n are the damping ratio and the undamped natural frequency respectively. The dynamic behavior of this second-order system can then be described in terms of two parameters ξ and ω_n. If $0<\xi<1$, the closed-loop poles are a pair of complex conjugates $(-\xi\omega_n \pm j\omega_n\sqrt{1-\xi^2})$ and lie in the left-half s-plane as shown in Fig. A.1. The system is then called underdamped, and the transient response is oscillatory [Ogat97]. If $\xi=1$, the system is called critically damped. Overdamped systems correspond to $\xi>1$.

For the second-order system specified by (A.1), the transient-response specifications such as the rise time, the peak time, the maximum overshoot and the settling time are determined by ξ and ω_n [Ogat97]. For example, the maximum overshoot decreases as ξ increases within the range between 0 and 1, and for a fixed value of ξ, the settling time decreases as ω_n increases.

For a higher order system, we adopt the dominant pole placement design approach [AsHa95]. A "dominant pole" is a pole closest to the origin on the real-axis direction in the s-

plane among all the closed-loop poles. Obviously the dominant pole determines the system performance. Therefore we can design a controller such that the pair of dominant poles can give rise to a desired damping ratio ξ and an undamped natural frequency ω_n, and also as long as all closed-loop poles are located at the left-half s-plane, the stability of the system is guaranteed according to well-known classical control theory. The dominant pole placement is a useful and popular design method since it gives predictable results and this approach is known to work well for most Single-Input Single-Output (SISO) plants by placing suitable dominant poles. This method results in better system closed-loop performance over those tuned by the classical Ziegler-Nichols rules [Ogat97]. When applying dominant pole placement to the SISO system, the controller parameters can be obtained analytically as to be shown in the design of the P_PP controller and the PI_PP controller in Chapter 3, and in the design of the ST_P_PP controller and the ST_PI_PP controller in Chapter 4.

Appendix B Preliminaries on $\mathcal{H}_\infty$ Optimal Control

In the eighties of the last century the $\mathcal{H}_\infty$ control problem became popular. In essence the $\mathcal{H}_\infty$ control problem is to find a controller that stabilizes a system and minimizes the $\mathcal{H}_\infty$-norm of a transfer function associated with the system. Ref. [Kwak93] is a good tutorial paper about the $\mathcal{H}_\infty$–optimization. Since $\mathcal{H}_\infty$ technique is relatively new, we would review in this appendix the $\mathcal{H}_\infty$–optimization problem and its solution to be used in Chapter 5 of the thesis.

B.1 The $\mathcal{H}_\infty$ Control and Three Forms of MSP

In 1979, one conference paper [Zame79] presented by Zames ignited the investigation of $\mathcal{H}_\infty$–optimization of control systems. In that paper, Zames considered the minimization of the $\mathcal{H}_\infty$-norm of the sensitivity function of a SISO (single-input-single-output) linear feedback system. Although the paper only dealt with some basic questions of classical control theory, it immediately caught significant attention in the control community.

The $\mathcal{H}_\infty$ norm of a stable scalar transfer function $f(s)$ is simply the peak value of $|f(j\omega)|$ as a function of frequency, namely[13],

$$\|f(s)\|_\infty \stackrel{\Delta}{=} \max_\omega |f(j\omega)| \quad .$$

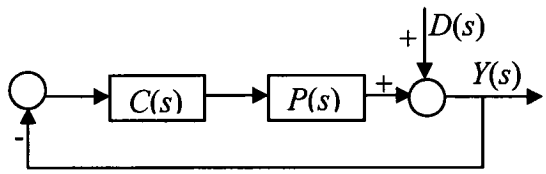

Fig. B.1: Basic SISO control system

$\mathcal{H}_\infty$-optimization of control systems deals with the minimization of the peak value of certain closed-loop frequency response functions. For an example, we first consider the sensitivity function of a feedback system. Fig. B.1 gives a basic SISO feedback system. $P(s)$

[13] Strictly speaking, we should here replace "max" (the maximum value) by "sup" (the supremum, the least upper bound). This is because the maximum may only be approached as $\omega \to \infty$ and may therefore not actually be achieved. However, for engineering purposes there is no difference between "sup" and "max". In the rest of the thesis, both terms are used.

and $C(s)$ are the plant transfer function and the controller (compensator) transfer function respectively. $D(s)$ represents a disturbance imposing on the system and $Y(s)$ is the control system output. From Fig. B.1 1 it follows that $Y=SD$ where

$$S = \frac{1}{1+PC} \ ,$$

is the sensitivity function of the feedback system. As the name implies, the sensitivity function characterizes the sensitivity of the control system output to disturbances. In the ideal case S should be zero.

The problem originally considered by Zames [Zame79] is to find a controller C that makes the closed-loop system stable and minimizes the peak value of the sensitivity function, i.e. to solve for

$$C_{\text{opt}} = \underset{C \text{ stabilizes } P}{\text{minimize}} \|S\|_\infty \ .$$

The justification of this problem is that if the peak value $\|S\|_\infty$ of the sensitivity function S is small, then the magnitude of S is necessarily small for all frequencies, therefore disturbances are uniformly attenuated over all frequencies. Minimization of $\|S\|_\infty$ is a worst-case optimization, since it amounts to minimizing the effect on the output of the worst disturbance.

A further investigation over the above problem reveals that the minimization of $\|S\|_\infty$ is not a useful design tool. The transfer function of every physical plant and controller decreases at high frequencies. This means that often the sensitivity S can be made small at low frequencies but eventually reaches the asymptotic value 1 for high frequencies. But how small S is at low frequencies is not reflected in the peak value while this is of paramount importance for the control system performance. Therefore, it is customary to introduce a frequency dependent weight function W_S and then we have the weighted sensitivity minimization problem, namely, we seek for an optimal controller C_{opt} such that

$$C_{\text{opt}} = \underset{C \text{ stabilizes } P}{\text{minimize}} \|W_S S\|_\infty \ .$$

Another way of understanding the $\mathcal{H}_\infty$-optimization is to use the concept of loop-shaping, i.e. shaping the closed-loop transfer functions, such as S, T, U. Here S is the sensitivity function as described above; T is the complementary sensitivity function, $T=1-S=PC/(1+PC)$, and it is the closed-loop transfer function from the reference input to the

control system output; U is input sensitivity function, $U=CS=C/(1+PC)$, and it is the transfer function from the reference input to the plant input. Here we consider S first. The sensitivity function S is a very good indicator of closed-loop performance. The main advantage of considering S is that we ideally want S small, it is sufficient to consider just its magnitude $|S|$; that is, we need not worry about its phase. Typical specifications in terms of S include:

1) The minimum bandwidth frequency ω_B (defined as the frequency where $|S(j\omega)|$ crosses 0.707 from below).
2) The system type, or alternatively the maximum steady-state tracking error[14], A.
3) The shape of S over selected frequency ranges.
4) The maximum peak magnitude of S, $\|S(j\omega)\|_\infty \leq M$.

The peak specification prevents amplification of noise at high frequencies, and also introduces a margin of robustness; $M = 2$ is a typical value. Mathematically, these specifications may be captured by an upper bound, $1/|W_S(s)|$, on the magnitude of S, where $W_S(s)$ is a weight function selected by the designer. Then the performance requirement becomes

$$|S(j\omega)| < 1/|W_S(j\omega)| , \quad \forall\omega \tag{B.1}$$

$$\Leftrightarrow \quad |W_S S| < 1 , \quad \forall\omega$$

$$\Leftrightarrow \quad \|W_S S\|_\infty < 1 .$$

The last equivalence follows from the definition of the $\mathcal{H}_\infty$ norm, and in words the performance requirement is that the $\mathcal{H}_\infty$ norm of the weighted sensitivity, $W_S S$, must be less than one. In the terminology of loop-shaping, Equ. (B.1) implies that the shape of the sensitivity function S is upper bounded by $1/|W_S(s)|$.

The specification $\| W_S S \|_\infty < 1$ does not allow us to specify the roll-off of the open loop gain, $L(s)=PC$. To do this one can make demands on another closed-loop transfer function, e.g., on the complementary sensitivity function T. For instance, one might specify an upper bound $1/|W_T(s)|$ on the magnitude of T to make sure that L rolls off sufficiently fast at high frequencies. Also, to achieve robustness or to restrict the magnitude of the plant input signals, one may place an upper bound $1/|W_U(s)|$ on the magnitude of the input sensitivity

[14] Note that S is also the transfer function from the reference input to the error signal (the controller input). Therefore S also determines the steady-state tracking error of the system.

function U. To combine these "mixed sensitivity" specifications, a "stacking approach" is usually used, resulting in the following overall specification

$$\|H\|_\infty = \max_\omega \bar{\sigma}(H(j\omega)) < 1 \quad , \qquad \text{where } H = \begin{bmatrix} W_S S \\ W_T T \\ W_U U \end{bmatrix} .$$

We here use the maximum singular value, $\bar{\sigma}(H(j\omega))$, to measure the size of the matrix H at each frequency. For SISO system, H is a vector and $\bar{\sigma}(H)$ is the usual Euclidean vector norm:

$$\bar{\sigma}(H) = \sqrt{|W_S S|^2 + |W_T T|^2 + |W_U U|^2} \quad .$$

After selecting the form of H and the weight functions, the $\mathcal{H}_\infty$ optimal controller C_{opt} can be obtained by solving the problem

$$C_{\text{opt}} = \underset{C \text{ stabilizes } P}{\text{minimize}} \|H(C)\|_\infty \quad .$$

By choosing different forms of H, we get different versions of the Mixed Sensitivity Problem (MSP). Usually we have the following three versions of MSPs.

B.1.1 *S/U* mixed-sensitivity optimization

In this case, $H = \begin{bmatrix} W_S S \\ W_U U \end{bmatrix}$, the S/U MSP is to find the optimal controller

$$C_{\text{opt}} = \underset{C \text{ stabilizes } P}{\text{minimize}} \left\| \begin{bmatrix} W_S S \\ W_U U \end{bmatrix} \right\|_\infty \quad . \tag{B.2}$$

B.1.2 *S/T* mixed-sensitivity optimization

Similarly, in the S/T MSP we seek

$$C_{\text{opt}} = \underset{C \text{ stabilizes } P}{\text{minimize}} \left\| \begin{bmatrix} W_S S \\ W_T T \end{bmatrix} \right\|_\infty \quad .$$

B.1.3 *S/T/U* mixed-sensitivity optimization

Choosing $H = \begin{bmatrix} W_S S \\ W_T T \\ W_U U \end{bmatrix}$, we get the 3-block $S/T/U$ MSP, namely one need to find C_{opt} such that

$$C_{\text{opt}} = \underset{C \text{ stabilizes } P}{\text{minimize}} \left\| \begin{bmatrix} W_S S \\ W_T T \\ W_U U \end{bmatrix} \right\|_\infty .$$

Obviously, for the MSP, the first step is to choose appropriate weight functions W_S, W_T, W_U. The discussion of the choices of weight functions is given in Section 5.2, 5.3 and 5.4.

B.2 Formulating Three MSPs in General Control Configuration

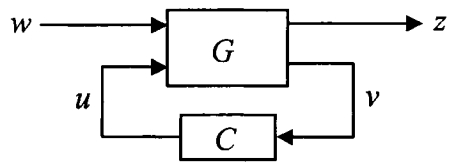

Fig. B.2: General control configuration for the case with no model uncertainty

Next we introduce the general control problem formulation that is needed in the state-space solution of the $\mathcal{H}_\infty$-optimization problem. A general control configuration is shown [Doyl83] in Fig. B.2, where G is the generalized plant model; C is the generalized controller; w is the exogenous inputs such as commands, disturbances and noise; z is the exogenous outputs, namely the so-called "error" signals to be minimized; v is the controller inputs including measured plant outputs, measured disturbances, etc; u is the control signals. The overall control objective is to minimize some norm of the transfer function from w to z, for example, the $\mathcal{H}_\infty$ norm. Then the $\mathcal{H}_\infty$-optimal controller design problem is:

- *Find a controller C which based on the information in v, generates a control signal u which counteracts the influence of w on z, thereby minimizing the closed-loop $\mathcal{H}_\infty$ norm from w to z.*

The system of Fig. B.2 is described by

$$\begin{bmatrix} z \\ v \end{bmatrix} = G(s) \begin{bmatrix} w \\ u \end{bmatrix} = \begin{bmatrix} G_{11}(s) & G_{12}(s) \\ G_{21}(s) & G_{22}(s) \end{bmatrix} \begin{bmatrix} w \\ u \end{bmatrix} , \tag{B.3}$$

$$u = C(s)v , \tag{B.4}$$

with a state-space realization of the generalized plant G given by

$$G \overset{s}{=} \left[\begin{array}{c|cc} A & B_1 & B_2 \\ \hline C_1 & D_{11} & D_{12} \\ C_2 & D_{21} & D_{22} \end{array}\right] . \tag{B.5}$$

The closed-loop transfer function from w to z is given [SkPo01] by the linear fractional transformation

$$z = F_l(G,C)w \ ,$$

where

$$F_l(G,C) = G_{11} + G_{12}C(I - G_{22}C)^{-1}G_{21} \ . \tag{B.6}$$

Therefore the $\mathcal{H}_\infty$-optimization problem becomes: seek an optimal controller C_{opt} such that

$$C_{\text{opt}} = \underset{\text{stabilizing } C}{\text{minimize}} \left\| F_l(G,C) \right\|_\infty \ . \tag{B.7}$$

The solution of the above problem will be considered in the next subsection. Here we formulate the three MSPs in the above general control configuration.

B.2.1 General control configuration for *S/U* MSP

Given the *S/U* MSP described by (B.2), we can formulate it in the context of general control problem so that (B.2) can be transformed to the form of (B.7).

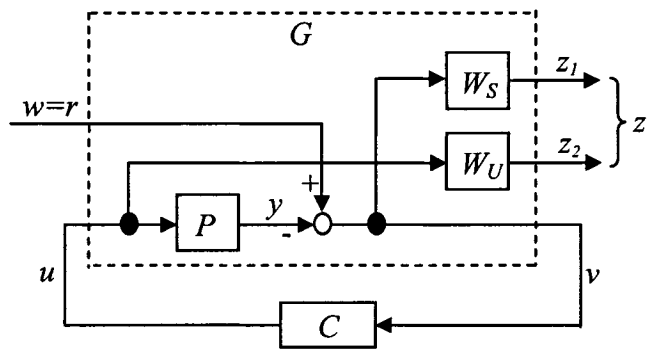

Fig. B.3: *S/U* mixed sensitivity optimization in the general control configuration

Fig. B.3 gives the general control configuration for the *S/U* mixed sensitivity optimization. Now we need to verify this. From Fig. B.3, we have

$$z_1 = W_S w - W_S P u \ ;$$

$$z_2 = W_U u \ ;$$

$$v = w - Pu \quad .$$

Therefore we could get

$$\begin{bmatrix} z \\ v \end{bmatrix} = \begin{bmatrix} G_{11} & G_{12} \\ G_{21} & G_{22} \end{bmatrix} \begin{bmatrix} w \\ u \end{bmatrix} \quad ,$$

where

$$z = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix} \quad , \quad G_{11} = \begin{bmatrix} W_S \\ 0 \end{bmatrix} \quad , \quad G_{12} = \begin{bmatrix} -W_S P \\ W_U \end{bmatrix} \quad ,$$

$$G_{21} = I \quad , \quad G_{22} = -P \quad .$$

Here I is the identity matrix, in the SISO case, $I = 1$. Then from (B.6), we could derive

$$F_l(G,C) = \begin{bmatrix} \dfrac{W_S}{1+PC} \\ \dfrac{W_U C}{1+PC} \end{bmatrix} = \begin{bmatrix} W_S S \\ W_U U \end{bmatrix} \quad .$$

This means Fig. B.3 does give the general control configuration for the *S/U* MSP.

B.2.2 General control configuration for *S/T* MSP

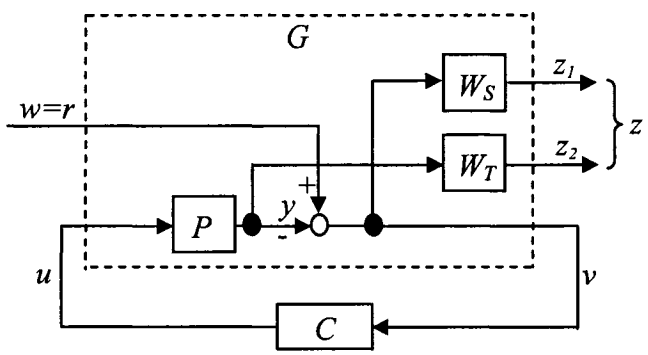

Fig. B.4: *S/T* mixed sensitivity optimization in the general control configuration

Likewise, Fig. B.4 shows the general control configuration for the *S/T* mixed sensitivity optimization. Since we have

$$G_{11} = \begin{bmatrix} W_S \\ 0 \end{bmatrix} \quad , \quad G_{12} = \begin{bmatrix} -W_S P \\ W_T P \end{bmatrix} \quad ,$$

$$G_{21} = I \quad , \quad G_{22} = -P \quad ,$$

and

$$F_l(G,C)=\begin{bmatrix}\dfrac{W_S}{1+PC}\\ \dfrac{W_TPC}{1+PC}\end{bmatrix}=\begin{bmatrix}W_SS\\ W_TT\end{bmatrix}\ .$$

B.2.3 General control configuration for *S/T/U* MSP

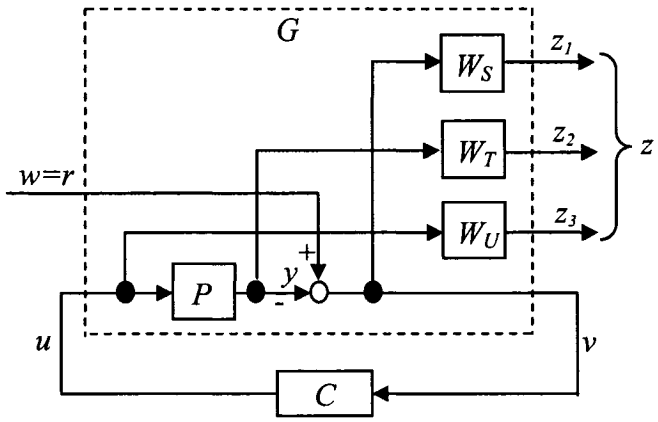

Fig. B.5: *S/T/U* mixed sensitivity optimization in the general control configuration

The general control configuration for *S/T/U* sensitivity optimization is shown in Fig. B.5, and we obtain

$$\begin{bmatrix}z\\ v\end{bmatrix}=\begin{bmatrix}G_{11} & G_{12}\\ G_{21} & G_{22}\end{bmatrix}\begin{bmatrix}w\\ u\end{bmatrix}\ ,$$

where

$$z=\begin{bmatrix}z_1\\ z_2\\ z_3\end{bmatrix}\ ,\quad G_{11}=\begin{bmatrix}W_S\\ 0\\ 0\end{bmatrix}\ ,\quad G_{12}=\begin{bmatrix}-W_SP\\ W_TP\\ W_U\end{bmatrix}\ ,$$

$$G_{21}=I\ ,\quad G_{22}=-P\ ,$$

and

$$F_l(G,C)=\begin{bmatrix}\dfrac{W_S}{1+PC}\\ \dfrac{W_TPC}{1+PC}\\ \dfrac{W_UC}{1+PC}\end{bmatrix}=\begin{bmatrix}W_SS\\ W_TT\\ W_UU\end{bmatrix}\ .$$

B.3 State-space Solution for the $\mathcal{H}_\infty$-Optimization Problem

In this subsection, we would discuss the solution for the $\mathcal{H}_\infty$-optimization problem defined by (B.7). Among the solutions available, the most general, widely used algorithm is based on the state-space solution summarized below [DoGl89].

Refer to (B.5), the following assumptions are typically made in the $\mathcal{H}_\infty$ problem:

(A1) (A, B_2, C_2) is stabilizable and detectable;

(A2) D_{12} and D_{21} have full rank;

(A3) $\begin{bmatrix} A - j\omega I & B_2 \\ C_1 & D_{12} \end{bmatrix}$ has full column rank for all ω.

(A4) $\begin{bmatrix} A - j\omega I & B_1 \\ C_2 & D_{21} \end{bmatrix}$ has full row rank for all ω.

(A5) $D_{11} = 0$ and $D_{22} = 0$;

Assumption (A1) is required for the existence of stabilizing controllers C, and assumption (A2) is sufficient to ensure the controllers are proper and hence realizable. Assumptions (A3) and (A4) ensure that the optimal controller does not try to cancel poles or zeros on the imaginary axis which would result in closed-loop instability. In assumption (A5), $D_{11} = 0$ makes G_{11} strictly proper; $D_{22} = 0$ makes G_{22} strictly proper. In the $\mathcal{H}_\infty$ problem, assumption (A5) significantly simplifies the algorithm formulas.

To get the solution of the $\mathcal{H}_\infty$ optimal controller defined by (B.7), we usually need two steps. First we solve the $\mathcal{H}_\infty$ sub-optimal control problem. That is, for a specified real number γ a stabilizing controller needs to be found for which $\|F_l(G, C)\|_\infty < \gamma$. In the next step, the algorithm can be used iteratively, reducing γ until the minimum is reached within a given tolerance and therefore the optimal controller is obtained.

The following gives the algorithm for solving the $\mathcal{H}_\infty$ sub-optimal control problem. For the general control configuration of Fig. B.2 described by equations (B.3)-(B.5), with the assumptions (A1) to (A5) holding, there exists a stabilizing controller C(s) such that $\|F_l(G, C)\|_\infty < \gamma$ if and only if

(i) $X_\infty \geq 0$ is a solution to the algebraic Riccati equation

$$A^T X_\infty + X_\infty A + C_1^T C_1 + X_\infty (\gamma^{-2} B_1 B_1^T - B_2 B_2^T) X_\infty = 0 \quad ,$$

such that Re $\lambda_i [A + (\gamma^{-2} B_1 B_1^T - B_2 B_2^T) X_\infty] < 0$, $\forall i$; and

(ii) $Y_\infty \geq 0$ is a solution to the algebraic Riccati equation

$$A Y_\infty + Y_\infty A^T + B_1 B_1^T + Y_\infty (\gamma^{-2} C_1^T C_1 - C_2^T C_2) Y_\infty = 0 \ ,$$

such that Re $\lambda_i [A + Y_\infty (\gamma^{-2} C_1^T C_1 - C_2^T C_2)] < 0$, $\forall i$; and

(iii) $\rho (X_\infty Y_\infty) < \gamma^2$.

All such controllers are then given by

$$C = F_l(C_K, Q) \ , \tag{B.8}$$

where

$$C_K(s) = \left[\begin{array}{c|cc} A_\infty & -Z_\infty L_\infty & Z_\infty B_2 \\ \hline F_\infty & 0 & I \\ -C_2 & I & 0 \end{array}\right]_s \ ,$$

$$F_\infty = -B_2^T X_\infty \ , \quad L_\infty = -Y_\infty C_2^T \ , \quad Z_\infty = (I - \gamma^{-2} Y_\infty X_\infty)^{-1} \ ,$$

$$A_\infty = A + \gamma^{-2} B_1 B_1^T X_\infty + B_2 F_\infty + Z_\infty L_\infty C_2 \ ,$$

and $Q(s)$ is any stable proper transfer function such that $\|Q\|_\infty < \gamma$. For $Q(s) = 0$, we obtain

$$C(s) = -Z_\infty L_\infty (sI - A_\infty)^{-1} F_\infty \ .$$

This is called the central controller and has the same number of states as the generalized plant $G(s)$.

Equ. (B.8) gives all the sub-optimal controllers for a specific value γ. If we desire a controller that achieves the minimized value γ_{min}, within a specified tolerance, then we can perform γ-iteration, e.g. a bisection search on γ until its value is sufficiently accurate. The above algorithm provides a test for each value of γ to determine whether it is less than γ_{min} or greater than γ_{min}.

Appendix C Preliminaries on Uncertainty Modeling and μ Analysis

In this Appendix, we review some preliminaries that are needed in the design of the MU_OPT controller in Chapter 6, including the uncertainty modeling, the general control configuration with uncertainty, analysis of NS, NP, RS and RP using μ-analysis and approximating the μ-optimal control problem with the $\mathcal{H}_\infty$ $S/T/U$ mixed sensitivity problem (MSP). Details can be found in [SkPo01].

C.1 Uncertainty Modeling

A control system is robust if it is insensitive to differences between the actual system and the model of the system which is used to design the controller. The differences are referred to as model uncertainty. The model uncertainty may have many sources. Some of these are (i) the parameters of the linear model are only known approximately or are simply in error; (ii) the parameters in the linear model may vary due to nonlinearities or changes in the operating conditions; (iii) at high frequencies even the structure and the model order is unknown; (iv) even when a very detailed model is available we may choose to work with a simpler (low-order) nominal model and represent the neglected dynamics as "uncertainty". Generally we group the above sources into two main classes:

1) Parametric uncertainty: the structure and the order of the model are known, but some of the parameters are uncertain.
2) Neglected and unmodeled dynamic uncertainty: the model is in error because of missing dynamics, usually at high frequencies, either through deliberate neglect or because of a lack of understanding of the physical process. Actually any model of a real system contains this kind of uncertainty.

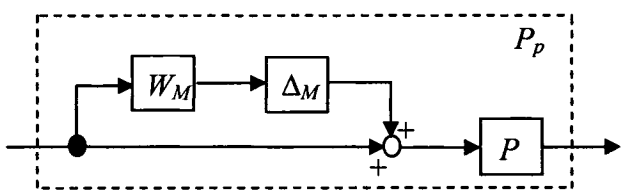

Fig. C.1: Plant with lumped multiplicative uncertainty

In the practical analysis, we usually combine the parametric uncertainty and the neglected / unmodeled uncertainty into one single lumped perturbation as shown in Fig. C.1, which shows the block diagram of a plant with lumped multiplicative uncertainty. In Fig. C.1, P is the nominal plant model with no uncertainty; P_p is the perturbed plant model with the lumped multiplicative uncertainty; W_M is the multiplicative uncertainty weight; Δ_M is the scaled multiplicative uncertainty, i.e. any stable transfer function which at each frequency is less than or equal to one in magnitude. Fig. C.1 shows that,

$$P_p(s) = P(s)(1 + W_M(s)\Delta_M(s)) \quad , \tag{C.1}$$

where $\Delta_M(s)$ satisfies: $|\Delta_M(j\omega)| \leq 1 \quad \forall \omega$, which is equivalent to $\|\Delta_M\|_\infty \leq 1$. In the robust controller design, the first step we need to do is the modeling of the uncertainty, i.e. to determine $W_M(s)$ in (C.1). The method we used to determine W_M is described in Section 6.2.1.

C.2 General Control Configuration with Uncertainty

In the robust performance analysis or controller synthesis for the system with uncertainty, we usually use a general control configuration, which will be introduced here.

Fig. C.2 shows the block diagram of a feedback system with multiplicative uncertainty. The performance weight $W_S(s)$ is also shown in the diagram. In Fig. C.2, W_M, Δ_M, P and W_S have been explained in Appendix C.1; C, w, z, v, and u are defined in Appendix B.2; u_Δ is the output of the scaled uncertainty; y_Δ is the input of the scaled uncertainty. Then we arrange Fig. C.2 in the form as shown in Fig. C.3. Therefore we could get the general control configuration with uncertainty as shown in Fig. C.4. We usually adopt this form for controller synthesis. However, if the controller is given and we need to analyze the uncertain system, we use the $N\Delta$-structure as shown in Fig. C.5, in which the controller C and the plant G are combined into N. It is well known that N is related to G and C by a lower LFT (linear fractional transformation)

$$N = F_l(G, C) = G_{11} + G_{12}C(I - G_{22}C)^{-1}G_{21} \quad .$$

Similarly, the uncertain closed-loop transfer function from w to z is related to N and Δ_M by an upper LFT

$$z = F_u(N, \Delta_M)w = (N_{22} + N_{21}\Delta_M(I - N_{11}\Delta_M)^{-1}N_{12})w \quad . \tag{C.2}$$

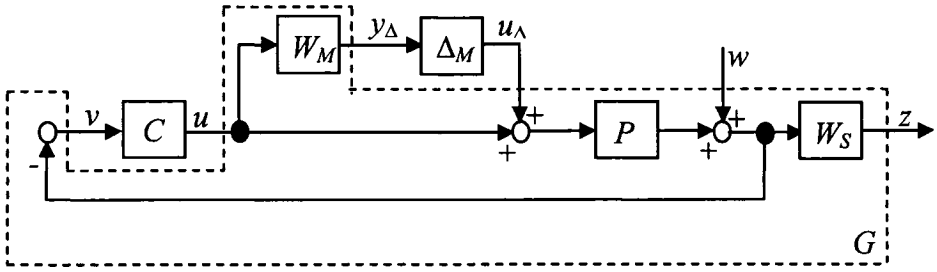

Fig. C.2: Feedback system with multiplicative uncertainty

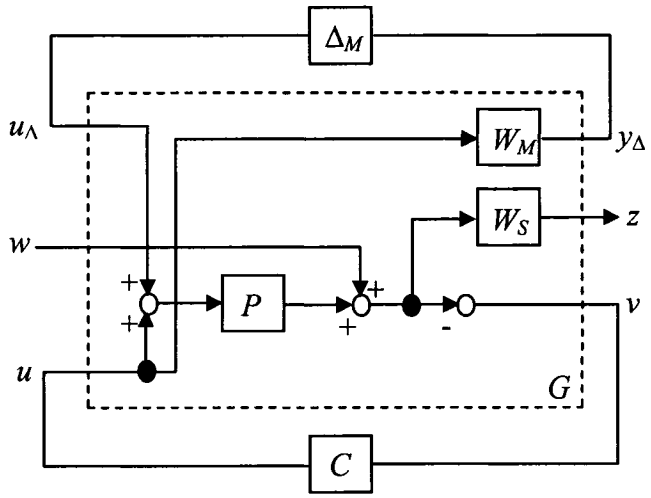

Fig. C.3: Rearranged block diagram of the feedback system with multiplicative uncertainty

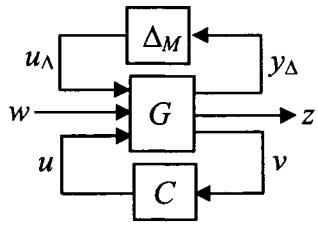

Fig. C.4: General control configuration with uncertainty (for controller synthesis)

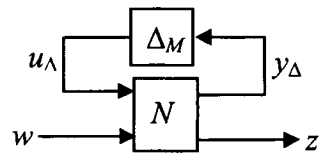

Fig. C.5: $N\Delta$-structure for robust performance analysis

C.3 Analysis of NS, NP, RS and RP Using μ-Analysis

In the robustness analysis for an uncertainty system, we need to check whether we have stability and acceptable performance for all plants in the uncertainty set:

1) *Robust* stability *(RS) analysis*: with a given controller C, we determine whether the system remains stable for all plants in the uncertainty set;
2) *Robust* performance *(RP) analysis*: if RS is satisfied, we determine how "large" the transfer function from exogenous inputs w to outputs z may be for all plants in the uncertainty set.

In terms of the $N\Delta$-structure in Fig. C.5 and described by (C.2), the requirements for stability and performance are defined as follows

$$\text{Nominal Stability (NS)} \overset{\text{def}}{\Leftrightarrow} N \text{ is internally stable;} \tag{C.3}$$

$$\text{Nominal Performance (NP)} \overset{\text{def}}{\Leftrightarrow} \|N_{22}\|_\infty<1 \text{; and NS;} \tag{C.4}$$

$$\text{Robust Stability (RS)} \overset{\text{def}}{\Leftrightarrow} F = F_u(N, \Delta_M) \text{ is stable } \forall \Delta_M, \|\Delta_M\|_\infty \le 1 \text{; and NS;} \tag{C.5}$$

$$\text{Robust Performance (RP)} \overset{\text{def}}{\Leftrightarrow} \|F\|_\infty<1, \forall \Delta_M, \|\Delta_M\|_\infty \le 1 \text{; and NS.} \tag{C.6}$$

Actually we can use structured singular value [Doyl82] to describe (C.4), (C.5) and (C.6). The structured singular value (denoted Mu or μ) is a function which provides a generalization of the singular value, $\bar{\sigma}$, and the spectral radius, ρ. It is defined as follows: *Find the smallest structured* Δ (*measured in terms of* $\bar{\sigma}(\Delta)$) *which makes* $\det(I-\Omega\Delta)=0$; *then* $\mu(\Omega) = 1/\bar{\sigma}(\Delta)$. Mathematically,

$$\mu(\Omega)^{-1} \overset{\Delta}{=} \min_{\Delta}\{\bar{\sigma}(\Delta) \mid \det(I - \Omega\Delta) = 0 \text{ for structured } \Delta\} \ .$$

Clearly, $\mu(\Omega)$ depends not only on Ω but also on the allowed structure for Δ. Therefore sometimes we explicitly use the notation $\mu_\Delta(\Omega)$.

Through the notion of structured singular value μ, definitions (C.3) to (C.6) can be proved to be equivalent to the following [SkPo01]:

$$\text{NS} \Leftrightarrow N \text{ is internally stable;} \tag{C.7}$$

$$\text{NP} \Leftrightarrow \bar{\sigma}(N_{22}) = \mu_{\Delta_S}(N_{22}) < 1, \ \forall \omega \text{; and NS;} \tag{C.8}$$

$$\text{RS} \Leftrightarrow \mu_{\Delta_M}(N_{11}) < 1, \ \forall \omega, \|\Delta_M\|_\infty \le 1 \text{; and NS;} \tag{C.9}$$

$$\text{RP} \Leftrightarrow \mu_{\tilde{\Delta}}(N) < 1,\ \forall \omega,\ \tilde{\Delta} = \begin{bmatrix} \Delta_M & 0 \\ 0 & \Delta_S \end{bmatrix},\ \|\Delta_M\|_\infty \le 1;\ \text{and NS;} \tag{C.10}$$

Here Δ_M is a block-diagonal matrix and its detailed structure depends on the uncertainty we are representing; Δ_S is always a full complex matrix and it is a fictitious uncertainty block representing the $\mathcal{H}_\infty$ norm performance specification. Therefore (C.8), (C.9) and (C.10) gives us a summary of μ-conditions for NP, RS and RP for an uncertainty system.

On rearranging the system in Fig. C.2 into the $N\Delta$-structure in Fig. C.5, we get,

$$N = \begin{bmatrix} W_M T & W_M CS \\ W_S SP & W_S S \end{bmatrix}. \tag{C.11}$$

where $T=CP(I+CP)^{-1}$, $S=(I+CP)^{-1}$.

Then for a SISO system, conditions (C.7)-(C.10) with N as in (C.11) become [SkPo01]

$$\text{NS} \Leftrightarrow \text{closed-loop system is nominally stable;} \tag{C.12}$$

$$\text{NP} \Leftrightarrow |W_S S| < 1,\ \forall \omega\text{; and NS;} \tag{C.13}$$

$$\text{RS} \Leftrightarrow |W_M T| < 1,\ \forall \omega\text{; and NS;} \tag{C.14}$$

$$\text{RP} \Leftrightarrow |W_S S| + |W_M T| < 1,\ \forall \omega\text{; and NS;} \tag{C.15}$$

C.4 Approximating μ-Optimal Control Problem with $\mathcal{H}_\infty$ $S/T/U$ MSP

From (C.15), it can be inferred that, the robust performance optimization, in terms of weighted sensitivity with multiplicative uncertainty for a SISO system, thus involves minimizing the peak value of robust performance index $\mu(N) = |W_M T| + |W_S S|$. Unfortunately at present there is no direct method to synthesize a μ-optimal controller. A closely related problem, which is easier to solve both mathematically and numerically, is to minimize the peak value ($\mathcal{H}_\infty$ norm) of the mixed sensitivity matrix

$$N_{mix} = \begin{bmatrix} W_S S \\ W_M T \end{bmatrix}.$$

Namely optimizing robust performance in terms of $\mu(N)$ is close to minimizing $\|N_{mix}\|_\infty$. In the case of TCP/AQM design, since we need to bound the plant input δp which is the variation of the probability of packet drop, we further approximate the robust performance optimization problem by introducing the term $W_U U$ in N_{mix},

$$N_{mix} = \begin{bmatrix} W_S S \\ W_M T \\ W_U U \end{bmatrix},$$

where W_U and U is the input sensitivity weight and input sensitivity transfer function respectively. Therefore minimizing $\mu(N)$ is approximated by solving the $\mathcal{H}_\infty$ $S/T/U$ mixed sensitivity problem (MSP), whose solution has been discussed in Chapter 5 and Appendix B.

Appendix D Nonlinear Model for the TCP/AQM System

This appendix provides the background of the linear model discussed in Chapter 2. The linear model presented in Section 2.2 is obtained from a nonlinear TCP/AQM model, which can be described by the following coupled, nonlinear differential equations [MiGo00], [HoMi01a]:

$$\dot{W}(t) = \frac{1}{R(t)} - \frac{W(t)W(t-R(t))}{2R(t-R(t))} p(t-R(t)) \ , \tag{D.1}$$

$$\dot{q} = \frac{W(t)}{R(t)} N(t) - C \ , \tag{D.2}$$

where $\dot{x}$ denotes the time-derivative of x and

W ≐ TCP window size (packets);
q ≐ queue length (packets);
R ≐ round trip time $= \frac{q}{C} + T_p$ (secs);
T_p ≐ propagation delay (secs);
C ≐ link capacity (packets/sec);
N ≐ load factor (number of long-lived TCP sessions);
p ≐ probability of packet drop.

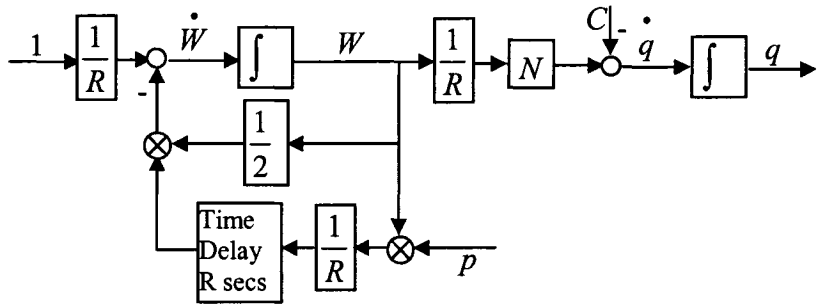

Fig. D.1: Block–diagram of the nonlinear dynamic model for TCP

The differential equations are illustrated in the block diagram of Fig. D.1 that highlights the TCP window-control and queue dynamics. Equ. (D.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. Equ. (D.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.

Appendix E Proof of the upper bounds of $|S|$ and $|U|$ in the $\mathcal{H}_\infty$ *S/U* MSP

This appendix presents the proof [SkPo01] of the upper bounds of $|S|$ and $|U|$ in the $\mathcal{H}_\infty$ *S/U* MSP.

Lemma: In the $\mathcal{H}_\infty$ *S/U* MSP defined by (B.2), if C_{opt} is used in the system, then $|S|$ and $|U|$ are upper bounded by $\gamma_{min}/|W_S|$ and $\gamma_{min}/|W_U|$ respectively, where γ_{min} is the optimal value of $\|H\|_\infty$ when C_{opt} is used.

Proof:

When C_{opt} is used, we have,

$$\begin{aligned}\gamma_{\min} &= \|H\|_\infty \\ &= \max_\omega \bar{\sigma}(H(j\omega)) \\ &= \max_\omega(\sqrt{|W_S S|^2 + |W_U U|^2}) \ ,\end{aligned}$$

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.