ON IMPROVING INTERNET QUALITY OF SERVICE USING A CONTROL THEORETIC APPROACH

By

XiaoLin Chang

A Thesis Submitted to
The Hong Kong University of Science and Technology
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy
in Computer Science Department

May 2005, Hong Kong
Authorization

I hereby declare that I am the sole author of the thesis.

I authorize the Hong Kong University of Science and Technology to lend this thesis to other institutions or individuals for the purpose of scholarly research.

I further authorize the Hong Kong University of Science and Technology to reproduce the thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Xiaolin CHANG
ON IMPROVING INTERNET QUALITY OF SERVICE USING A CONTROL THEORETIC APPROACH

by

Xiaolin CHANG

This is to certify that I have examined the above Ph.D thesis and have found that it is complete and satisfactory in all respects, and that any and all revisions required by the thesis examination committee have been made.

M. K. R. Jogesh

Dr. Jogesh K. Muppala, Supervisor

Lionel M. NI

Prof. Lionel M. NI, Department Head

Department of Computer Science
30 May 2005
Acknowledgements

I am taking this opportunity to thank individually all those who have assisted me during my Ph.D study.

First and foremost, I would like to express my deepest gratitude to my supervisor, Dr. Jogesh K. MUPPALA. I wish to thank him for consistent guidance, inspirational encouragements, patience, and kindness throughout my study.

I would like to thank my committee members, Prof. John.C.S LUI, Prof. Mounir HAMDI, Dr. Brahim BENSAOU, Dr. Shueng-Han CHAN, and Dr. Li QIU for serving on the examination committee and for their great effort on my thesis. I especially thank Prof. Mounir HAMDI and Dr. Brahim BENSAOU for their invaluable discussions and comments in my proposal defense.

I would like to thank Prof. Christopher V. Hollot, Prof. Yossi Chait, and Prof. Vadim I. Utkin for their invaluable helps during my research. I would like to thank HKUST students, Xiaodong ZHAO, Jijie XU, Xiaojun HEI, Alex Jun ZHANG for many helpful discussions.

I would like to thank communication tutor, Mrs. Shauna Dalton who helped me to improve my written-ups.

I would like to thank to my office colleagues and roommates in HKUST. Their enjoyable friendship is memorable.

At last but not least, I would like to thank my parents for their sacrifice and love for so many years; thank my son for his understanding, who has been a big boy now; thank my husband for his continuous love and companionship throughout.
TABLE OF CONTENTS

Title Page i
Authorization Page ii
Signature Page iii
Acknowledgements iv
Table of Contents v
List of Figures viii
List of Tables x
Abstract xi

Chapter 1 Introduction 1
Chapter 2 Background and Related Work 5
2.1 A Generic PID-type structure 5
2.1.1 Basic elements in feedback control system 5
2.1.2 Performance metrics 5
2.1.3 Control theory based design methodology: PID 8
2.1.4 Tuning method 9
2.2 Congestion control in current TCP/IP wired networks 9
2.3 Active Queue Management 11
2.4 Bandwidth assurance in AF-based DiffServ networks 13
2.4.1 The AF-based DiffServ networks 13
2.4.2 Intelligent traffic conditioner 14
2.5 Wireless networks 16
2.5.1 Wireless systems – characteristics 16
2.5.2 Wireless Local Area Networks 17

Chapter 3 Analysis and Design Intelligent Traffic Conditioner in AF-based DiffServ networks 22
3.1 Introduction 22
3.2 Local stability analysis of an under-subscribed AF-based DiffServ network 23
3.2.1 A generic NPI-type controller structure for designing intelligent traffic conditioners 23
3.2.2 System model 25
3.2.3 Local stability analysis 27
3.2.4 An illustrative example 36
3.2.5 Relations between network parameters and $k_{PL,ACT_j}$ 38
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>3.3</td>
<td>Analysis of some existing mechanisms</td>
<td>39</td>
</tr>
<tr>
<td>3.3.1</td>
<td>The rules of tuning controller gains</td>
<td>40</td>
</tr>
<tr>
<td>3.3.2</td>
<td>Analysis of some existing mechanisms</td>
<td>42</td>
</tr>
<tr>
<td>3.3.3</td>
<td>ARM</td>
<td>43</td>
</tr>
<tr>
<td>3.4</td>
<td>Variable-structure PI controller for adapting CIR threshold</td>
<td>45</td>
</tr>
<tr>
<td>3.4.1</td>
<td>The VS-ACT mechanism</td>
<td>45</td>
</tr>
<tr>
<td>3.4.2</td>
<td>Design considerations</td>
<td>46</td>
</tr>
<tr>
<td>3.5</td>
<td>Simulation results</td>
<td>49</td>
</tr>
<tr>
<td>3.5.1</td>
<td>Static network scenarios: under-subscribed</td>
<td>52</td>
</tr>
<tr>
<td>3.5.2</td>
<td>Static network: exact-subscribed</td>
<td>56</td>
</tr>
<tr>
<td>3.5.3</td>
<td>Dynamic networks</td>
<td>57</td>
</tr>
<tr>
<td>3.5.4</td>
<td>The importance of upper bound and lower bound on CIR_{threshold}</td>
<td>60</td>
</tr>
<tr>
<td>3.6</td>
<td>Discussion and conclusion</td>
<td>62</td>
</tr>
<tr>
<td>Chapter 4</td>
<td>A Stable Queue-based Adaptive Controller for AQM</td>
<td>64</td>
</tr>
<tr>
<td>4.1</td>
<td>Introduction</td>
<td>64</td>
</tr>
<tr>
<td>4.2</td>
<td>Stability condition for queue-based adaptive PI controller</td>
<td>65</td>
</tr>
<tr>
<td>4.2.1</td>
<td>The existing linearized AIMD/AQM system model</td>
<td>65</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Motivation for varying gains according to queue length error</td>
<td>66</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Stability conditions for PI controller with varying gains</td>
<td>66</td>
</tr>
<tr>
<td>4.3</td>
<td>Implementation of a queue-based adaptive PI controller</td>
<td>72</td>
</tr>
<tr>
<td>4.3.1</td>
<td>Description of the implementation</td>
<td>72</td>
</tr>
<tr>
<td>4.3.2</td>
<td>Selecting parameters: (k_{\text{min}}, k_{\text{max}}, k_{\text{inc}}, k_{\text{in}})</td>
<td>72</td>
</tr>
<tr>
<td>4.3.3</td>
<td>Discussions of (k_0, k_1, k_2)</td>
<td>75</td>
</tr>
<tr>
<td>4.4</td>
<td>Performance evaluation</td>
<td>77</td>
</tr>
<tr>
<td>4.4.1</td>
<td>Varying FTP flows</td>
<td>79</td>
</tr>
<tr>
<td>4.4.2</td>
<td>Varying round-trip time</td>
<td>81</td>
</tr>
<tr>
<td>4.4.3</td>
<td>Varying bottleneck link capacity</td>
<td>83</td>
</tr>
<tr>
<td>4.4.4</td>
<td>Multiple links</td>
<td>85</td>
</tr>
<tr>
<td>4.5</td>
<td>Summaries</td>
<td>86</td>
</tr>
<tr>
<td>Chapter 5</td>
<td>The Effects of AQM on The Performance of Assured Forwarding Service</td>
<td>89</td>
</tr>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>89</td>
</tr>
<tr>
<td>5.2</td>
<td>Related work</td>
<td>90</td>
</tr>
<tr>
<td>5.3</td>
<td>The implementations of some two-level AQMs</td>
<td>91</td>
</tr>
<tr>
<td>5.3.1</td>
<td>Two-level PI controller</td>
<td>92</td>
</tr>
<tr>
<td>5.3.2</td>
<td>Other two-level AQMs</td>
<td>93</td>
</tr>
<tr>
<td>5.4</td>
<td>Simulation configuration</td>
<td>93</td>
</tr>
<tr>
<td>5.5</td>
<td>Performance evaluation under standard traffic conditioner</td>
<td>97</td>
</tr>
<tr>
<td>5.5.1</td>
<td>Dumbbell network topology: effects on the attainment of excessive bandwidth</td>
<td>97</td>
</tr>
<tr>
<td>5.5.2</td>
<td>Dumbbell network topology: effects on the achievement of bandwidth assurance</td>
<td>100</td>
</tr>
<tr>
<td>5.5.3</td>
<td>Different CIRs</td>
<td>100</td>
</tr>
<tr>
<td>5.5.4</td>
<td>Different propagation delays</td>
<td>103</td>
</tr>
<tr>
<td>5.5.5</td>
<td>Multiple bottleneck links: effect on achievement of bandwidth assurance</td>
<td>103</td>
</tr>
<tr>
<td>5.6</td>
<td>Performance evaluation under intelligent traffic conditioner</td>
<td>104</td>
</tr>
<tr>
<td>5.6.1</td>
<td>Simulation 1: in the network scenario of Section 5.5.4 (Different RTTs)</td>
<td>105</td>
</tr>
<tr>
<td>5.6.2</td>
<td>Simulation 2: Different # of micro-flows in each aggregate</td>
<td>106</td>
</tr>
<tr>
<td>5.7</td>
<td>STAVQ</td>
<td>107</td>
</tr>
</tbody>
</table>
List of Figures

Figure 2.1 Block diagram of feedback control system ........................................ 6
Figure 2.2 Block diagram of TCP/AQM system .................................................. 6
Figure 2.3 Block diagram of AF-based DiffServ system ..................................... 6
Figure 2.4 DiffServ architecture ........................................................................ 14
Figure 2.5 Marking algorithm in TSW2CM ...................................................... 14
Figure 2.6 WLAN ............................................................................................. 19
Figure 3.1 The combined Adjusting-Algorithm/AQM AF-based DiffServ network .................................................. 24
Figure 3.2 The combined PI-ACT/RIO AF-based DiffServ network .................... 28
Figure 3.3 Simple feedback system ................................................................... 29
Figure 3.4 Block diagram of AQM networks with active PI loops (the perturbation block indexes correspond to those in the set J) .......................................................... 29
Figure 3.5 Magnitude Bode plot of (\( \hat{P} \Delta \)) .............................................. 38
Figure 3.6 Magnitude Bode plot of \( \Delta \) ......................................................... 38
Figure 3.7 Magnitude Bode plot of \( \hat{P} \) ......................................................... 38
Figure 3.8 Goodput and \( CIR_{\text{thresh}} \) variations of A1-A3 ................................ 38
Figure 3.9 Bottleneck link queue length variation .............................................. 39
Figure 3.10 The i-th AF-feedback-loop system with VS-ACt controller as the Adjusting-Algorithm ................................................................. 47
Figure 3.11 Illustration of function of \( e \) over time ........................................... 47
Figure 3.12 Network topology ......................................................................... 50
Figure 3.13 Static under-subscribed network --- Impact of \# of micro-flows .... 53
Figure 3.14 Static under-subscribed network --- Impact of packet size ............ 54
Figure 3.15 Static under-subscribed network --- Impact of target rate ............. 54
Figure 3.16 Static under-subscribed network --- Impact of RTT ...................... 55
Figure 3.17 \( \text{SumCIR}_{\text{thresh}} \) ........................................................................... 56
Figure 3.18 Static exact-subscribed network .................................................... 57
Figure 3.19 Average Goodput variation of A1-A7 ............................................ 59
Figure 3.20 Goodput and \( CIR_{\text{thresh}} \) variations of A1 .................................. 59
Figure 3.21 Varying number of micro-flows in aggregates ................................ 61
Figure 3.22 Performance of A1 and A3 ............................................................ 62
Figure 3.23 Performance of A9 and A10 ............................................................ 62
Figure 4.1 AIMD/AQM system using Q-SAPI controller .................................. 66
Figure 4.2 Equivalent to System in Figure 4.1 ................................................. 68
Figure 4.3 Equivalent to System in Figure 4.1 ................................................. 71
Figure 4.4 Equivalent to System in Figure 4.3 ................................................. 71
Figure 4.5 \( k_p(t) \) and \( k_i(t) \) algorithms .............................................................. 73
Figure 4.6 Dropping/marking probability algorithm ......................................... 73
Figure 4.7 Modified Nyquist trajectory and Popov straight line for \( T_1(s) \) ........ 73
Figure 4.8 Modified Nyquist trajectory and Popov straight line for \( T_2(s) \) ........ 74
Figure 4.9 \( f(r) \) for \( k_{\text{incr}}^p =1/8958, k_{\text{incr}}^i =1/292000, q_1=q_2=0.99999 \) .......... 76
Figure 4.10 Variation of \( k_p(t) \) versus \( |e(t)| \) ......................................................... 76
Figure 4.11 Network topology with single bottleneck link ................................ 78
Figure 4.12 Dynamic forward FTP flows with \( q_{\text{rel}} =100 \) packets ................. 80
Figure 4.13 Dynamic forward FTP flows with \( q_{\text{rel}} =0 \) packets ..................... 81
Figure 4.14 Dynamic RTT ............................................................................. 82
Figure 4.15 Dynamic capacity in [10Mbps,50Mbps] ......................................... 84
Figure 4.16 Dynamic capacity in [10Mbps,100Mbps] ....................................... 85
List of Tables

Table I  Control gains in each scheme................................................................. 46
Table II  Attributes of aggregates........................................................................ 51
Table III Schemes................................................................................................... 51
Table IV Notations and parameters settings............................................................ 78
Table V  Common Variables....................................................................................... 92
Table VI  Schemes .................................................................................................. 97
Table VII Bottleneck Link Throughput (C=20.0 Mbps)........................................... 101
Table VIII Bottleneck Link Throughput (C=20.0 Mbps)......................................... 104
Table IX  Bottleneck Link Throughput (C=20.0 Mbps)........................................... 106
Table X  Average Goodput....................................................................................... 119
On Improving Internet Quality of Service Using a Control Theoretic Approach

by

Xiaolin CHANG

Department of Computer Science
Hong Kong University of Science and Technology

Abstract

Rapid development in telecommunication technologies and ever-growing network-user demands have made network congestion a critical problem in today’s Internet. Congestion not only leads to significant network-centric performance degradation such as low link utilization, but also damages the user-centric Quality-of-Service (QoS). In addition, the increasingly heterogeneous network environments plus varying and unpredictable traffic conditions suggests that any mechanism without adaptability hardly be able to deliver QoS requirements.

This thesis considers the issue of congestion control and two QoS issues brought up by congestion in various network infrastructures including wired networks and wireless LANs. The two QoS issues are: (i) bandwidth assurance in the Assured Forwarding (AF) service in Differentiated Services (DiffServ) wired networks; and (ii) application level fairness between downlink and uplink flows in an infrastructure WLAN using the DCF mode. This thesis explores using a control theoretic approach to analyze and design mechanisms for these problems based only on the local knowledge.

First we describe a generic controller structure and discuss its features. Then, we use it to design a stable queue-based adaptive AQM mechanism for the current Internet. We aim to alleviate two tradeoffs: (i) between fast transient response and stability, and (ii) between fast transient response and small steady state error. Stability analysis of the closed-loop nonlinear system is conducted, providing guidelines for designing the algorithm and selecting control parameters.

The current Internet is a best-effort network and most AQM schemes focus on providing QoS assurance from a network-centric standpoint. The DiffServ approach has
been proposed as a scalable mechanism in order to provide QoS. However, the AF service in this approach fails providing bandwidth assurance under certain conditions. In this thesis, we instantiate the generic controller structure in the AF service framework and analyze the features of the existing ingress-based mechanisms for improving bandwidth assurance among AF flows. Based on the analysis, we propose a simple but robust controller for this problem. Extensive simulations are carried out to validate the analysis and demonstrate that the proposed mechanism outperforms the mechanisms proposed in the literature over a wide range of network dynamics.

Congestion control and fairness issues are not only prominent for the Internet infrastructure when wired backbones and centralized control entities are available, but are also essential for various applications that use wireless technologies for flexible data communication. Therefore, this dissertation extends the research methodologies and technologies for Internet congestion control and QoS problems to wireless LANs. We explore the feasibility of combining AQM schemes with the underlying MAC protocol design in WLANs to provide fairness for traffic flows traversing between wired and wireless domains.
CHAPTER 1

INTRODUCTION

The recent past has witnessed rapid and tremendous technology development in the packet-switched wired and wireless networks. There is a strong consensus today that IP will be the foundation of next-generation networking [60]. As packet-switched wired and wireless networks continue to evolve in popularity, size, and diversity, there appear ever-growing demands for meeting Quality-of-Service (QoS) requirements of network users and applications. However, network congestion creates serious challenges to the provision of manageable, stable and reliable services to meet the QoS demands in packet-switched networks, such as the current Internet. Currently, the network providers offer these services by keeping the network utilization low, which in the long term may not be a cost-effective solution. Congestion control is an important network requirement in packet-switched networks. Handling congestion efficiently to improve network utilization, while providing a satisfactory level of service to the users, is known to be a practical, but challenging problem [58].

Congestion control in the current Internet mainly depends on the end-to-end mechanisms at the transport layer. Transport Control Protocol (TCP) congestion control mechanisms in current TCP/IP networks are triggered by packet loss, which is detected by timeout or triple duplicate acknowledgment. When packets are dropped only due to router buffer overflow, the ability of TCP congestion control mechanisms is degraded. In addition, TCP congestion control is vulnerable to various aggressive and congestion-insensitive flows. While some recent research have been carried out to replace the existing TCP protocol with better congestion control mechanisms [40], it is not trivial, if not impossible, to deploy such mechanisms at each end system. It would be better if Internet routers participate in congestion control.

Algorithms that the routers employ to detect, measure and control congestion are called Active Queue management (AQM) schemes. They aim to improve the overall system's performance from a network-centric perspective by providing congestion
information to sources, which in turn adjust their rates. However, an AQM scheme without adaptability hardly meets the QoS requirements in dynamic networks. Although some adaptive AQM schemes have been developed such as [17] and [78], they provide no theoretic analysis. Consequently the system stability in the dynamic networks is unknown. The performance of an unstable system can severely and persistently diverge from the desired performance so as to cause system malfunctioning and even complete system failure.

The current Internet offers only best-effort services. Moreover, most AQM schemes focus on providing QoS assurance from a network-centric standpoint such as improving link utilization and stabilizing router queue length. The Differentiated Services (DiffServ) [8] approach has been proposed as a scalable mechanism to overcome this insufficiency. Assured Forwarding (AF) [35] Per Hop Behavior (PHB) is one of the forwarding mechanisms standardized by the Internet Engineering Task Force (IETF). The AF service aims to offer different forwarding assurances to different customers based on their profiles. Several recent research studies within the current DiffServ framework, such as in [11], [69] and [72], have brought forth the failure of the AF service in providing bandwidth assurance under certain conditions. Several intelligent traffic conditioners have been proposed to address this failure. However, most of them are evaluated through simulations and no theoretic analysis is carried out to explain the reason when these conditioners fail to provide bandwidth assurance.

Congestion control and fairness issues are not only prominent for the Internet infrastructure when wired backbones and centralized control entities are available, but also are very essential for various applications that use wireless technologies for flexible data communication. The rapid growth of the Internet and the Internet protocol-related applications shift the wireless system toward supporting a wide variety of data services. The characteristics of wireless media results in that the wireless bandwidth is less than the bandwidth in wired domains. In the saturated infrastructure Wireless LANs (WLANs) using the Distributed Coordination Function (DCF) mode, the channel capacity available for the wireless interface of the access point (AP) may be even smaller. The channel capacity disparity between the wired interface and the wireless interface of the AP leads to congestion at the AP in the infrastructure WLANs using the DCF mode. This
congestion issue subsequently compromises the provision of fairness to traffic flows traversing between wired and wireless domains. However, the existing mechanisms, which handle congestion and QoS issues in wired networks, cannot be directly applied to the wireless domain without modification. The main reasons include (i) the unique characteristics of wireless transmission such as the shared nature of the wireless media, error-prone channel, limited energy supply and scarce capacity; (ii) unlike in the wired networks where congestion control is mostly provided by network layer and transport layer protocols, these issues in wireless networks are also coupled with the underlying the media access control (MAC) protocol design.

The above discussions motive the research presented in this dissertation. We focus on the congestion issue and two QoS issues brought up by congestion in various network infrastructures including wired networks and wireless WLAN. The two QoS issues are: (i) bandwidth assurance in the AF service in DiffServ wired networks; (ii) application level fairness between downlink and uplink flows in the infrastructure wireless LAN using DCF mode. This dissertation explores using a control theoretic approach to analyze and design mechanisms for attacking these problems. Note that control theoretic approaches have been widely used to analyze and design mechanisms to improve the performance of various software systems [5]. In addition, this dissertation explores whether these issues can be effectively handled based only on the local knowledge, gathered at the routers, and the remedial schemes initiated locally.

The contributions of this dissertation are as follows:

- **AQM for network-centric QoS**:
  Designing a stable queue-based adaptive controller for congestion control in the wired networks. Stability analysis of the non-linear system is conducted.

- **Bandwidth assurance in the AF-based DiffServ networks**:
  1. Instantiating a nonlinear Proportional-Integral-derivative (PID)-type structure in the AF service framework. Using this structure to analyze the existing ingress-based mechanisms in the literature.
  2. Analyzing the system stability and then deriving the sufficient conditions for system stability.
(3) Designing an explicit-state-based self-tuning controller for improving bandwidth assurance.

(4) Carrying out an experimental study of the impact of existing AQMs on the performance of the AF service and on the ability of the intelligent traffic conditioners.

- **Congestion and QoS in wireless LANs:**
  1. Proposing a modified queue-based AQM with ECN-marking to manage the interface queue (IFQ) in order to control congestion at the AP in the WLANs.
  2. Exploiting the information of IFQ queue length and channel state to improve application level downlink and uplink fairness in the infrastructure WLANs using DCF mode.

The rest of this thesis is organized as follows. In Chapter 2, we present the background and some related work, including the introduction of the control theoretic design approach used in this thesis. In Chapter 3, we use the approach to analyze and design ingress-based mechanisms for improving bandwidth assurance in the AF-based DiffServ networks. In Chapter 4, we apply the approach to design a stable queue-based adaptive mechanism for AQM. In Chapter 5, we investigate the effects of different AQMs on the performance of the AF service and on the ability of ingress-based mechanisms. In Chapter 6, we shift to wireless LANs. We explore how to control congestion and explore how to improve downlink and uplink fairness no matter whether there exist non-adaptive flows. Summary and future work are given in Chapter 7.
CHAPTER 2
BACKGROUND AND RELATED WORK

2.1 A Generic PID-type structure

2.1.1 Basic elements in feedback control system

A typical structure of the elementary type of feedback control systems is illustrated in Figure 2.1, including a plant to be controlled, a controller, and a sensor. The plant is system's inherent behavior. For example, in the TCP/AQM networks, TCP dynamics plus queue dynamics comprise the plant; an AQM scheme is the controller. Figure 2.2 and Figure 2.3 are instantiations of Figure 2.1, respectively in the TCP/AQM networks and the AF-based DiffServ networks.

In a controller, there are three basic control-related variables, the reference value, manipulated variable and controlled variable. The controlled variable is the system output, which is to be measured and needs to be controlled. The sensor measures the value of the controlled variable when there is no way to obtain this value directly. The reference value represents the target value of the controlled variable. The manipulated variable is the system attribute, which is dynamically modified by the controller so as to affect the value of the controlled variable. The controller input, error, is defined as the difference between the reference value and the current value of the controlled variable, i.e., \( \text{error} = \text{reference value} - \text{current value of controlled variable} \). In such a closed-loop system, the system periodically monitors and compares the controlled variable with the reference value to determine the error; the controller changes the value of the manipulated variable to control the system output by using the control algorithm and the value of error.

2.1.2 Performance metrics

In a closed-loop system, the performance metrics must characterize important transient and steady state properties of a system in terms of its controlled variables. The steady state is defined as a state when the controlled variable stays within e% of its
reference value. From the control theory point of view, a system transits from the steady state to the transient state when the controlled variable deviates significantly from its steady state value in response to changes in its run-time conditions. After a period in the transient state, the system may settle down to a new steady state and the controlled variable converges to the vicinity of a new value.

The performance metrics are as follows. Here we note that in different contexts, some of the following metrics are not considered.

- **Stability**: A system is Bounded-Input-Bounded-Out stable if its controlled variables are always bounded for bounded performance references and disturbances. Stability
is a necessary condition for achieving the desired performance reference value. It is especially an important requirement because a poorly designed controller can overreact to performance errors and drive a system to unstable situations. The performance of an unstable system can severely and persistently diverge from the desired performance so as to cause system malfunctioning and even cause complete system failure. Currently there is not a theory to predict the network behavior when it loses stability [38].

- **Transient state response**: Describe the responsiveness and efficiency of mechanisms in reacting to changes in run-time conditions.

- **Settling time**: The time it takes the system to settle down to a steady state from the start of a transient state. The settling time represents how fast the system can regain desired performance after a change in its run-time condition.

- **Overshoot**: The maximum amount that a controlled variable overshoots its reference divided by its reference, i.e., \( C_0 = (C_{\text{m}} - C_r)/C_r \) where \( C_{\text{m}} \) is the maximum value of the controlled variable during its transient state. Overshoot characterizes the worst-case transient performance degradation of a system. A system may require a low overshoot. For example, in the AF-based DiffServ networks, the aggregate consists of HTTP flows. A large overshoot may lead to buffer overflow and then most HTTP flows experience timeout.

- **Steady state error**: The difference between the average value of a controlled variable in the steady state and its reference value. The steady state error characterizes how precisely the system can enforce desired performance in steady state. The steady state error will depend on the type of input (step, ramp, etc) as well as the system type (0, 1, or II). Steady-state error analysis is only useful for stable systems. It is meaningless to discuss steady state error for an unstable system.

- **Sensitivity**: Sensitivity describes the robustness of the system with regard to traffic characteristics and network resource variations. It reflects the relative change of a controlled variable in steady state with respect to the relative change of a system parameter. For example, assuming the controlled variable is the queue length, the system's sensitivity with respect to the bursty traffic represents how significantly the changes in network parameters affects the queue length.
2.1.3 Control theory based design methodology: PID

The PID-type controller structure has been widely used due to its simplicity and no requirement of a precise analytical model of the system being controlled. Eq.(2.1) gives the continuous-time Nonlinear Proportional-Integral-Derivative control (NPID) controller structure [1].

\[ p(t) = K_p(\cdot) \times e(t) + K_i(\cdot) \times \int e(t) + K_d(\cdot) \times \dot{e}(t) \]  

(2.1)

Eq.(2.1) consists of three components: (i) Proportional control \( K_p(\cdot) \times e(t) \), which delivers an output which is proportional to the size of the error signal. It aims to reduce the rise time and reduce, but never eliminate, the steady state error. (ii) Integral control \( K_i(\cdot) \times \int e(t) \) aims to eliminate the steady state error, but may slow down the transient response. (iii) Derivative control \( K_d(\cdot) \times \dot{e}(t) \), which aims to prevent overshoot and undershoot of the controller variable and aims to restore controller variable rapidly to the reference value if there is a sudden change in system conditions. \( K_p(\cdot), K_i(\cdot) \) and \( K_d(\cdot) \) are all controller gains, respectively representing the proportional gain, the integral gain, and the derivative gains; \( p(t) \) and \( e(t) = q(t) - q_0 \) are the controller output and input, respectively.

Implementing a PID controller requires considering four major inter-related aspects: (i) identify the three basic control-related variables, the reference value, manipulated variable and controlled variable; (ii) determine the controller components; note that Eq.(2.1) can be tailored as any combination of the three components; (iii) design the controller gain adjusting algorithms; (iv) set the control parameters in the adjusting algorithms. When controller gains are not all constant, we refer to the controller as adaptive controller; otherwise, it is a fixed-gain controller. When designing a fixed-gain controller, (iii) and (iv) merge into “choose the values of the controller gains”. Controller gains may be functions of system states or be functions of plant parameters.
2.1.4 Tuning method

Designing an efficient NPID-type controller requires proper settings of controller gains in order to produce short settling time, small steady state error and low sensitivity in the dynamic networks. A good understanding of the plant behavior (i.e., the cause-effect relationship between controlled and manipulated variables) is extremely helpful in designing the rules for control actions to be taken. In this thesis, we adjust \( K_p \) and \( K_i \) by using the system-state-based self-tuning method, which is an implementation of the direct adaptive control approach [3]. In this method, \( K_p \) and \( K_i \) are designed as functions of the system output such as queue length. Note that \( K_p \) and \( K_i \) can also be tuned by applying the indirect adaptive control approach [3], which use the estimated plant parameters to update the controller gains. This approach has been applied to improve the performance of the various software systems, such as in [49][55][77][82]. The discussion about the advantages and disadvantages of two approaches is beyond this thesis.

2.2 Congestion control in current TCP/IP wired networks

This section reviews the interaction between congestion, buffering in routers, and the traffic types that share a network.

The Internet is a collection of small networks connected by store-and-forward devices, called routers. Each router has multiple bi-directional links that it could direct the packet along. Using information from a routing table, the router decides the next link, which the packet will cross. Each of these links is attached to a line card (LC), which is responsible for processing the packet. Each link has a finite, fixed capacity. If the amount of data that needs to be forwarded along a link arrives at a faster rate than what the link can transmit, the link is overloaded and is said to be congested. Congestion challenges the provision of the manageable, stable and reliable services to meet the QoS demands. It may waste network resources due to the dropped packets when queues overflow, result in long delays in data delivery, and even lead to possible congestion collapse [37].

Congestion may be transient or persistent. Packets may arrive in a bursty manner. That is, a large number of packets arrive nearly simultaneously, in a group, followed by a gap during which few packets arrive. Bursty traffic is a common phenomenon in the
Internet. It may lead to transient congestion. This kind congestion can be remedied by providing buffering in the router to allow packets for a given out-bound link to be stored briefly before being forwarded.

Alternatively, the congestion may be persistent, i.e. the arriving load is consistently exceeding the capacity of the link. Although router buffering can address transient congestion, it is not a viable solution to persistent congestion. Persistent congestion has traditionally been addressed with the cooperation from end systems. Persistent congestion control could “open-loop congestion control” or “closed-loop congestion control”. Congestion prevention is “open-loop congestion control”. In the kind control, the end systems need to negotiate with the network providers before sending packets. The negotiation often involves how to distribute buffer space and bandwidth to arriving packets, how many packets should be admitted into the network, and how to discard packets when buffers are full. Any negotiation aims to make sure that no more traffic than what the network can handle will be injected into the network. Therefore no congestion will occur. Generally, after the initial negotiation is done, the end-system and the routers act independently and the end-system gets no information from the network about the current traffic and network status. This makes the system inflexible in a dynamic Internet environment and the add-on control mechanisms such as admission control, traffic shaping and policing functionalities also complicate the router design.

In the case of “closed-loop congestion control”, the end-system needs to get feedback information from the network, such as the current congestion status. The TCP/IP networks are such kind system. TCP congestion control mechanisms regard a packet loss as congestion indication. A packet loss is detected by timeout or triple duplicate acknowledgment (ACK) packets. Whenever congestion is detected, TCP congestion control mechanisms at the end system reduce the injected load in order to match the available capacity of the network and then alleviate the congestion. When packets are dropped only due to router buffer overflow, the ability of TCP congestion control mechanisms is degraded. As most of the Internet traffic uses TCP as the underlying transport protocol, it is desirable to design an intelligent router queue management mechanism to assist the end-systems in dealing with congestion.
2.3 Active Queue Management

Internet routers use Active Queue management (AQM) mechanisms to provide support for congestion control and Quality of Service (QoS) [61]. Possible QoS requirements include low packet loss and short queuing delay, high link utilization, less bias against bursty sources [21], and fair sharing among competing connections. An AQM mechanism must define how to detect congestion and how to inform the involved end-hosts. The information used to detect incipient congestion can be per-aggregate or per-flow arriving rate, average/instantaneous queue length, or a combination of arrival rate and queue length. Besides using packet drops to implicitly inform the sources of congestion, an AQM mechanism can explicitly signal end-systems of incipient network congestion by marking the Explicit Congestion Notification (ECN) [67] bit in the IP header. ECN-marking has a significant impact on packet loss rate and link utilization [50].

Since Random Early Detection (RED) [21] was introduced into Internet routers, many improvements to RED and many new AQM mechanisms have been proposed in the past few years in order to overcome the weakness of RED. Some are designed based on heuristics. The disadvantage of designing in an ad-hoc way is that very little is known about the reasons why these AQMs work and very little explanation is given when they fail. Recently, systematic design methodologies have been applied to analyze existing AQM mechanisms and design new AQM mechanisms. Different methodologies interpret the congestion control problem in different ways. Optimization-based approaches are based on formulating the TCP/AQM dynamics from an optimization standpoint as a convex program. In this program the aggregated source utility is maximized subject to the bottleneck link capacity constraints [41][42][45][51]. The Adaptive Virtual Queue (AVQ) [46] scheme is a modified primal algorithm for this optimization problem, aimed to obtain the optimal source rate. The Random Exponential Marking (REM) [4] algorithm, which tries to obtain the optimal congestion measures, is a modified dual algorithm for the optimization problem. Both the primal algorithms and the dual algorithms have dynamics at one end (user end for the primal algorithm and link end for the dual algorithm) and static adaptation at the other. The authors in [53] propose Exponential-RED, which is a primal-dual algorithm. Optimization-based approaches largely lead to steady state equilibrium, but may not work well in the transient states.
Essentially, TCP/AQM system is a feedback control system. Recognizing this, several AQMs based on control theoretic approaches have been developed. The authors in [2] design an AQM scheme by directly applying the basic ideas of feedback control theory. However, they do not give details of how to select control parameters and they present no theoretic performance analysis. It is due to the lack of an analytical model of TCP/AQM dynamics. The authors in [61] develop a fluid-flow model of TCP/AQM dynamics and in [33] develop the simplified linear-time-invariant model. A variant of this model proposed in [17], considers using ECN-marking to notify the sources of congestion and considers the presence of non-adaptive flows. Although these models roughly describe the packet level effects, they enable the theoretic analysis of the system performance and enable the selection of control parameters.

PID-type approach has been used to design AQM mechanisms and to analyze various existing AQM mechanisms for Internet congestion control [17]. Based on the linearized fluid-based TCP/AQM model [33], the authors in [32] developed a Proportional-Integral (PI) controller for AQM. This PI-controller aims not only to improve its responsiveness to the TCP/AQM dynamics by means of proportional control using the instantaneous queue length rather than using the EWMA queue length [68], but also to stabilize the router queue length around the target queue length by means of Integral control regardless of the load level. In order to improve robustness, that is the ability of stabilizing over a wide range of network dynamics, the fixed-gain PI-type AQM mechanisms select control gains for the worse case. The result is that they cannot achieve both satisfactory transient response and small steady state error simultaneously over a wide range of network dynamics. The main reasons include (i) the lag caused by the integral control; (ii) the fixed controller gains, which are directly related to network parameters; then the control system exhibits either a fast transient response with large queue fluctuations or a sluggish response with small queue fluctuations when the network or traffic conditions are changing. In order to reduce the impact of the first factor, some authors propose to abandon integral control such as in [29] and [30] or add derivative (D) control such as in [64] and [17]. These improvements are still fixed-gain controllers. The authors in [64] give a survey of advances in AQMs, including these PI(D) controllers.
Designing adaptive AQMs has been attracting the interest of researchers recently. Controller gains can be tuned by applying the indirect adaptive control approach, which uses the estimated network parameters to update the controller gains. Network parameters mentioned in this thesis include the maximum RTT, number of active long-lived TCP flows, and link capacity. The mechanisms in [77] and [82] are such examples. In addition, controller gains can be adjusted by using the system-state-based self-tuning method. In this thesis, we refer to as control parameters those parameters in the algorithms for adjusting controller gains.

Some system-state-based adaptive mechanisms for AQMs have been developed. Adaptive PI (API) [17] and Loss Ratio based RED (LRED) [78] are two examples. API achieves adaptability by assuming the current dropping probability as the new steady state dropping probability. LRED achieves adaptability by assuming the low-pass filtered measured packet loss ratio as the new steady state dropping probability.

2.4 Bandwidth assurance in AF-based DiffServ networks

2.4.1 The AF-based DiffServ networks

Figure 2.4 shows the general framework of a DiffServ network. In this framework, the routers are divided into two categories, core routers (simple and high speed) and edge routers (stateful and intelligent). The Time Sliding Window Two Color Marker (TSW2CM) [26] is one of the packet marking algorithms proposed to work with the AF service. Figure 2.5 outlines the algorithm of TSW2CM. Under TSW2CM, the edge nodes monitor all the incoming packets and mark their DiffServ Codepoint (DSCP) to in-profile (IN) or out-profile (OUT) according to the Service level Agreement (SLA). The core routers do not have any per-flow state and differentiate packets based solely on the DSCP marking of the packet performed at the edge routers.

Some definitions used in the following are given now. CIR represents the Committed Information Rate, defined in the SLA. The marking threshold \(CIR_{\text{Thresh}}\) of a priority level for an aggregate is the marking rate, determining the allowed maximum low-pass
filtered arrival rate of the packets belonging to the priority levels no lower than this priority level. In Time Sliding Window Three Color Marker (TSW3CM) [26], there are two marking thresholds, set to CIR and Peak Information Rate (PIR) respectively. In TSW2CM, there is only one marking thresholds, set to CIR.

2.4.2 Intelligent traffic conditioner

Ever since Clark proposed the AF-based service framework by using RED with in/out (RIO) mechanism [9], extensive performance studies of the AF service in this framework have been carried out. In order to improve the performance of AF service, some researchers design intelligent AQM schemes at core routers, for example in [11] and in the references cited in [70]. In this thesis we only consider intelligent schemes designed at
ingress routers to improve bandwidth assurance. Moreover we assume that no intelligent dropping/scheduling schemes are implemented at core routers. This assumption is consistent with the DiffServ approach of pushing the complexity to the network edges such that the core of the network is simple and scalable. Note that the mechanisms employed at ingress routers and the intelligent AQM schemes at core routers are complementary and can be used in conjunction with each other.

According to the number of levels of drop preference in an AF class, there are two kinds of traffic conditioners: (i) two-color based and (ii) three-color based. Three-color based conditioners, such as TSW3CM, are proposed in order to improve the fair sharing of excess bandwidth. Through simulations, the authors in [27] conclude that utility of three levels of drop precedence in a traffic class depends on the traffic load, the sum of target rates and the available link capacity.

Intelligent traffic conditioners at ingress routers have been proposed in [12][14][19][25][31][44][54][62]. They only use the knowledge gathered at the ingress router to improve bandwidth assurance in the context of the AF-based services. Through appropriately marking the packets as IN/OUT at the ingress routers, these mechanisms improve bandwidth assurance by indirectly influencing the rate, which congested routers allocate to flows/aggregates. Before we proceed, we define some terms used subsequently in the thesis. We refer to a flow/aggregate as unsatisfied when its bandwidth assurance is not achieved; otherwise it is considered satisfied. We refer to a flow/aggregate as conditionally-satisfied when its bandwidth assurance is achieved by increasing $CIR_{Thresh}$ larger than $CIR$. The essence of the remedies suggested in [12][14][19][25][31][44][54][62] is to increase the allowed maximum low-pass filtered arriving rate of IN traffic of the unsatisfied flow/aggregate so that a larger amount of IN traffic of this flow/aggregate is injected into the domain. Then, the performance loss caused by the dropped/ECN-marked low priority packets is compensated. Some authors, for example in [19][31][54][62] implement this by incorporating TCP flow characteristics (such as RTT, packet size, Retransmission Time-Out) into the computation of marking probabilities. These remedies can be applied only to individual flows or require that all the micro-flows in an aggregate be identical.
Some intelligent schemes for aggregates based on aggregate information have been proposed, such as in [12][14][25][44]. Usually, this aggregate information is the low-pass filtered average arriving rate, which is computed at ingress routers either periodically or upon a packet arrival. In order to reduce the large performance fluctuation and reduce the sensitivity to control parameter settings, the authors in [44] propose the *Memory-based Marking* (MBM). MBM adjusts marking probabilities (defined as \(mp\)) upon a data packet arrival by using the current average arriving rate and the previous average arriving rate. Note that the average arriving rate is computed also upon the arrival of a packet. In [25] the authors propose a *Packet Marking Engine* (PME, employed at routers). PME uses the periodically estimated average arriving rate as the decision-making factor in the process of adjusting the marking probability periodically. The authors in [12] develop an *Adaptive CIR Threshold* (ACT). ACT adapts the marking threshold (defined as \(CIR_{\text{Thresh}}\)) periodically. By using classical linear control theory, the authors in [14] propose a fluid flow model for the dynamics in the AF-based DiffServ network and develop the *Active Rate Management* (ARM). ARM regulates the token bucket rate at ingress routers to guarantee the minimum bandwidth requirement. Token bucket rate is a kind of marking threshold. ARM also uses the periodically estimated average arriving rate as the decision-making factor in adjusting the token bucket rate periodically. Note that \(mp\) of a satisfied aggregate in PME or the token bucket rate of a satisfied aggregate in ARM may be decreased to zero. However, ACT uses CIR as the lower bound of \(CIR_{\text{Thresh}}\).

### 2.5 Wireless networks

#### 2.5.1 Wireless systems – characteristics

So far all the discussions have focused on conventional wired networks. The recent success of wireless technologies has boosted the development of wireless networks. Most current networking protocols were designed mainly for wired networks. Many assumptions that make these protocols efficient in wired networks are no longer true in wireless networks. The performance of these networking protocols degrades dramatically in wireless environments.
The performance degradation stems from the unique characteristics of wireless media as compared to wired media. Wireless communications come in many forms and span a large range of capacity and distances. The wireless world includes short range communications (e.g. bluetooth, HomeRF), local mobility wireless hot spots such as WLANs (e.g 802.11b, ETSI HyperLAN2), "long-thin" wide-area cellular networks (e.g GSM, GPRS, IS-95, IS-95B and UMTS-3G) and finally the "long-fat" satellite communication networks.

In addition to its wide range of forms, wireless media is also essentially different from traditional wired media. First, the available bandwidth of the wireless media is limited and shared by many users, and the overall throughput is therefore much lower than that of wired media. A typical category 5 unshielded twisted-pair (UTP) cable can deliver 1 Gbps bandwidth in Gigabit networks, but a typical wireless WAN in GSM is designed for 9600 bps per user. In wired networks, laying more cables yields more bandwidth, so the bandwidth of wired media can be expanded infinitely. On the other hand, the bandwidth of wireless media is limited by the available radio spectrum. The allocation of the spectrum is normally controlled by government agencies. While radio spectrum experts are looking for ways to encode more bits in the spectrum, the available spectrum remains limited. In this sense, the wireless bandwidth cannot be expanded infinitely. Second, wireless media is shared by two direction flows. Competition for the shared medium causes access contention. If not coordinated properly, the access contention can result in low efficiency and unpredictable channel access delay. Third, wireless media is prone to transmission errors. Environmental conditions and terrestrial obstructions and reflections, noise, channel interference and fading lead to high and unpredictable error rates. Fourth, extensive error correction and error detection mechanisms lead to high delay or delay variation.

2.5.2 Wireless Local Area Networks

Due to the ease of deployment and the moderate cost, there has been a surge in the popularity of WLANs all over the world in recent years. WLANs provide an effective means of achieving wireless data connectivity in offices, homes, campuses and other local environments and are expected to be an integral part of next-generation wireless
communication networks. The most popular standard for wireless LANs is IEEE 802.11 [36]. The IEEE 802.11 WLAN specification standardizes physical layer and medium access control (MAC) sub-layer implementations.

There are two operational modes for IEEE 802.11 WLANs: ad hoc mode and infrastructure mode. The IEEE 802.11 Medium Access Control (MAC) layer specifies two different mechanisms for contention resolution: Coordination Function (PCF) and Distributed Coordination Function (DCF). PCF is a centralized polling based access scheme. DCF defines a distributed access algorithm based on the carrier sense multiple accesses with contention avoidance (CSMA/CA). DCF can be used in both infrastructure and ad hoc wireless LANs. Both these contention resolution mechanisms attempt to ensure a reasonably fair opportunity of accessing the medium to all participating hosts.

The architecture of a typical wireless local area network (WLAN) is shown in Figure 2.6. The key components in this architecture include the mobile nodes and the base stations. The mobile node can be a mobile telephone, a PDA, or a laptop computer with a wireless modem. The base station is a stationary collection of hardware components that communicate with nearby mobile nodes through radio media. It is responsible for relaying packets between mobile nodes, or between the mobile nodes and wired networks. Therefore, it is often called an “access point” (AP).

2.5.2.1 Downlink congestion

The characteristics of wireless media results in that the wireless bandwidth is less than the bandwidth in wired domains. In the saturated infrastructure Wireless LANs (WLANs) using the Distributed Coordination Function (DCF) mode, the channel capacity available for the wireless interface of the AP may be even smaller. The channel capacity disparity between the wired interface and the wireless interface of the AP makes the access point a significant network congestion point in the downstream direction. Similar to the wired networks, this congestion affects the provision of fairness.

So far only a handful of research (see, for example, [80] and the references cited therein) has investigated using AQM to control this congestion. They only consider the network scenarios of pure downlink flows. Downlink (uplink) flows are referred to as those flows whose sources are wired (mobile) nodes and receivers are mobile (wired)
nodes. Downlink (uplink) packets represent packets sent from wired (mobile) nodes to mobile (wired) nodes.

2.5.2.2 Fairness in WLANs

The widespread deployment of WLANs during the recent past results in the increased number of applications with distinct QoS requirements. IEEE 802.11 offers only best-effort services, not guaranteeing any service level to users/applications. Issues of QoS arise whenever something is shared.

For different types of applications and different wireless forms, there are different requirements for QoS. There have been many proposals on QoS provision in wireless networks. We only consider the fair WLAN bandwidth distribution between uplink and downlink flows in the infrastructure WLANs using the DCF mode. By fair distribution, we mean that the amount of traffic received by the receivers of uplink flows is equal to that of downlink flows. We refer to this fair (unfair) WLAN bandwidth distribution as uplink/downlink fairness (unfairness). This unfairness is mainly caused by the equal probability of accessing the wireless medium to all competing nodes in the legacy DCF. This unfairness is exacerbated in the case of TCP flows because of the greedy closed loop control nature of TCP [65].

19
Some fair-scheduling algorithms (see the references cited in [18] and [47]) have been proposed to allocate network resource fairly to each competing flow in proportion to a given flow weight. However, they attempt either for resource allocation among downlink flows or for the fairness among STAs in a WLAN. The unfair sharing of bandwidth between the uplink and the downlink still remains [47].

The authors in [16] and [65] consider the scenarios where there are only TCP flows. They proposed different methods, deployed above the MAC layer, to limit the TCP flow rates in order to achieve uplink/downlink fairness. The authors in [65] assume that the AP knows the number of active TCP flows \( n \). Then they proposed to feedback the information of \( \lceil (\text{buffer size})/n \rceil \) via ACK packets to the senders in order to achieve uplink/downlink fairness. As long as the number of active flows is known, this simple method works perfectly. The authors in [16] assume that the knowledge of the maximum achievable WLAN bandwidth is available. Then an IP level rate-limiter is used to limit the rate of uplink flows. One problem of this mechanism is the low WLAN bandwidth utilization when the maximum sending rate of all wired senders is less than half of maximum achievable WLAN bandwidth.

Some mechanisms for this issue have also developed at the MAC layer in order to accommodate more kinds of traffic. The authors in [47] propose the mechanism FAIR for the mixed networks. In FAIR, when the AP successfully receives an MAC ACK frame from the mobile nodes, it checks whether the downlink side achieves the specified throughput. If not, the AP enters the downlink compensation access (DCA) mode. If the downlink side achieves the specified throughput or no ACK frame is transmitted on the wireless channel, AP enters the DCF mode. In DCA, (i) the downlink frames (including Data packets from wired senders and ACK packets for mobile senders) are transmitted using point inter-frame space (PIFS) following the previous ACK frame; (ii) the handshake mechanism of RTS and CTS is ignored.

Similar to FAIR, the authors in [7] ignore the type of the used transport protocol. Since my research only considers the uplink/downlink fairness, we ignore the part of the mechanism in [7] for achieving good fairness among downlink flows. My research only considers the part for improving uplink/downlink fairness, which is referred to as DyCW. In DyCW, the AP periodically informs the mobile nodes about the number \( N \) of active
mobile nodes and the minimum contention window of AP (\(CW_{\text{minAP}}\)). When the active mobile nodes receive this information, they set their own \(CW_{\text{min}}\) to \((N \times (CW_{\text{minAP}} + 1) - 1)\).
CHAPTER 3

ANALYSIS AND DESIGN INTELLIGENT TRAFFIC CONDITIONER
IN AF-BASED DIFFSERV NETWORKS

3.1 Introduction

In Section 2.4.2, we have mentioned some ingress-based mechanisms have been proposed to improve the bandwidth assurance in the AF-based DiffServ networks. Note that MBM, PME and ACT utilize feedback control in an ad hoc manner. To our knowledge, ARM is the first mechanism that has been designed based on feedback control theory. It consists of a fixed-gain PI controller and a low-pass filter. In addition, to our knowledge, no work explains why the ad hoc mechanisms work and very little explanation is given when they fail to provide bandwidth assurance under some circumstances. This motivates the work in this chapter.

This chapter assumes that ingress routers have only output queues and only core routers may be congested. Discussions about the implementation of ingress-based mechanisms in combined input and output queuing (CIOQ) switches are given in Section 3.6. We use routers and switches interchangeably in this chapter.

The rest of this chapter is organized as follows. In Section 3.2 we first instantiate the generic NPI-type controller structure in the AF context. Then, we analyze the local stability of an under-subscribed AF-based DiffServ network. In Section 3.3 we first discuss the tuning rules for controller gains and then analyze some existing intelligent traffic conditioners. In Section 3.4, we propose a simple but robust controller, Variable-Structure Adaptive CIR Threshold (VS-ACT), for improving bandwidth assurance. In Section 3.5, extensive simulation studies are carried out in the various static and dynamic networks to investigate our analysis and the performance of VS-ACT. Section 3.6 gives conclusions. By ending this section, we summarize the features of VS-ACT

- Adjusting CIR threshold rather than PIR threshold. Choosing the value of PIR (the lower bound of \( PIR_{\text{Thresh}} \)) in the three-color model is critical to exhibit the capability of three-color based markers in improving bandwidth assurance. Too
small PIR may not work effectively. Too large PIR may cause best-effort sources to
suffer. Best-effort traffic and FIFO scheduling will continue to have their place also,
due to their fundamental simplicity and "good enough" performance as a simple
method is still deployed. Thus, we choose to adjust CIR\text{thresh}. The standard traffic
conditioner mentioned in the following is referred to TSW2CM.

- **Model-independent self-tuning PI controller.** By model-independent self-tuning,
  we mean that when the varying ranges of the controller gains are determined, the
  controller gains of the PI controller are adjusted on-line based on the system states
  rather than based on network parameters.

- **Using local knowledge and using remedies initiated locally.** Thus there is no
  additional overhead added to the network links and core routers. Local knowledge
  is defined as the knowledge gathered at ingress routers. Another advantage of using
  local information and local remedies is that only the aggregates in trouble are
  involved. Then some wrong control decision, such as caused by the local wrong
  measurement, can't lead to serious adverse impacts. Meanwhile the wrong action
  could quickly be rectified and then the adverse impact could be eliminated quickly.

### 3.2 Local stability analysis of an under-subscribed AF-based DiffServ
network

In this section, we first give a generic NPI-type controller structure for designing
intelligent conditioners. Then we carry out the local stability analysis of an under-
subscribed AF-based DiffServ network, which provides guidelines for designing tuning
rules.

#### 3.2.1 A generic NPI-type controller structure for designing intelligent traffic
conditioners

Figure 3.1 gives the closed-loop architecture of a combined Adjusting-
Algorithm/AQM AF-based DiffServ network, where each bold inner loop is called an AF-
feedback-loop. This loop is invoked at every sampling instant. Each dotted box denotes
an ingress router. From the control theory point of view, the Adjusting-Algorithm is the
controller and "Aggregate i Dynamics" (0 ≤ i ≤ n, n is the number of ingress routers) is the
Figure 3.1 The combined Adjusting-Algorithm/AQM AF-based DiffServ network

plant in the AF-feedback-loop. The sensor measures the arrival rate. The Adjusting-Algorithm employed at ingress routers aims to improve bandwidth assurance by increasing the amount of IN priority traffic based only on the local knowledge.

We define the controller input at time instant $k$ as $e(k) = CIR - r_s(k)$, where CIR is the reference value, that is, the target rate defined in SLA. $r_s$ is defined as the low-pass filtered arrival rate due to the bursty nature of the network traffic and other perturbations. Now the key issue is defining the controller output. Some mechanisms such as PME and MBM improve bandwidth assurance by adjusting $mp$; others such as ARM and ACT improve bandwidth assurance by adjusting $CIR_{\text{Thresh}}$. Since the essential idea of all the mechanisms is to improve the amount of IN priority traffic, the controller output should completely determine the allowed maximum low-pass filtered arrival rate of IN priority traffic. When the controller output is defined as $mp$, this maximum rate is still affected by $r_s$. However, when the controller output is $CIR_{\text{Thresh}}$, the controller output completely determines this maximum rate. Thus, we choose $CIR_{\text{Thresh}}$ as the controller output in our general NPI-type controller structure. Correctly defining the controller output benefits the analysis of a mechanism.
Now we consider designing the controller structure. The authors in [33] discuss the limitations of applying a pure proportional controller for AQM. Similar limitations exist when a pure proportional controller is applied to adjust $CIR_{\text{Thresh}}$. Adding Integral control can alleviate these limitations. Eq.(3.1) gives a continuous-time NPI-type controller structure

\[
CIR_{\text{Thresh}}(t) = K_p(\cdot) \times e(t) + K_i(\cdot) \times \int e(t)
\]

(3.1)

Eq.(3.1) is the combination of the proportional control action $CIR_{\text{Thresh}}(t) = K_p(\cdot) \times e(t)$ and the integral control action $CIR_{\text{Thresh}}(t) = CIR_{\text{Thresh}}(t-1) + K_i(\cdot) \times \int e(t)$. We do not use Derivative (D) control because the network traffic is bursty and Derivative control may amplify the noise.

3.2.2 System model

This subsection presents the local stability analysis of an under-subscribed AF-based DiffServ network where (i) there are $n$ heterogeneous aggregates, each consisting of $N_i$ identical long-lived TCP connections; (ii) the $i$-th ingress router uses the TSW profiler to provide two-level edge coloring and uses a PI-type marker with fixed-gains to adjust the marking threshold $CIR_{\text{Thresh}}^i$; (iii) RIO is used as AQM at the core router with an infinite and non-emptying buffer. Without loss of generality, we assume that each aggregate is served by a separate ingress router. The traffic of all aggregates feeds into a core router with link capacity $C$ and queue length denoted by $q(t)$. Our starting point is the linearized model for the standard AF-based DiffServ network and the method of analyzing stability proposed in [15]. For simplicity, we assume the dropping probability of IN traffic at congested routers is zero. Before continuing, we first introduce the notations that are used in the following. The subscript $i$ refers to the $i$-th aggregate, from 1 to $n$.

- $C$: link capacity (packets/sec).
- $N_i$: the number of micro-flows in the $i$-th aggregate.
- $R_i$: the round-trip delay of a micro-flow in the $i$-th aggregate (second).
- $p_i$: dropping/marketing probability of red traffic.
- $q$: instantaneous queue length (packets).
- $W_i$: window size of a micro-flow in the $i$-th aggregate (packet).
- $T_{pi}$: is the average propagation delay of the $i$-th aggregate.
\( x_i \): the sending rate of the \( i \)-th aggregate.
\( CIR_{Th}^i \): the marking threshold of the \( i \)-th aggregate.
\(|*|\): the magnitude of \(*\).

A linearized model of the under-subscribed AF-based DiffServ network around the equilibrium point \((q^*, W_i^*, p_i^*, (CIR_{Th}^*)^*)\) is described by

\[
\begin{align*}
\delta W_i(s) &= \frac{\partial g_i}{\partial CIR_{Th}^i} \delta CIR_{Th}^i(s) + \frac{\partial g_i}{\partial p_i} e^{-sR_i} \delta p_i(s) \\
\delta q(s) &= \sum_{i=1}^{n} \frac{\partial W_i}{\partial q} \delta W_i(s)
\end{align*}
\]

(3.2)

where

\[
\begin{align*}
\dot{W}_i(t) &= g_i(q, W_i, p_i, CIR_{Th}^i) \\
\dot{q}(t) &= f(q, W_i, p_i, CIR_{Th}^i) \\
\frac{\partial f}{\partial q} &= \sum_{i=1}^{n} x_i \\
\frac{\partial f}{\partial W_i} &= \frac{N_i}{R_i} \\
\frac{\partial g_i}{\partial CIR_{Th}^i} &= \left( \frac{CIR_{Th}^i}{2N_i} - \frac{CIR_{Th}^i}{N_iW_i^2} \right) + \frac{W_i}{R_i} \frac{W_i}{R_i} \\
\frac{\partial g_i}{\partial p_i} &= \frac{1}{R_i} + \frac{CIR_{Th}^i}{N_iW_i} + \frac{W_iCIR_{Th}^i}{2N_i} - \frac{W_i^2}{2R_i} \\
\delta W_i(t) &= W_i(t) - W_i^* \\
\delta q(t) &= q(t) - q^* \\
\delta p_i(t) &= p_i(t) - p_i^* \\
\delta CIR_{Th}^i(t) &= CIR_{Th}^i(t) - \left( CIR_{Th}^* \right)^*
\end{align*}
\]

The details of the model can be found in [15]. The equilibrium point \((q^*, W_i^*, p_i^*, (CIR_{Th}^*)^*)\) satisfies the following equations
\[
0 = 1 - \left(1 - \frac{CIR_{th}}{x_i}\right) p_t - 0.5 \left(1 - \frac{CIR_{th}}{x_i}\right) p_t W_i^2 \\
R_i(t) = \frac{T_{pi}}{C} + \frac{g}{C} \\
0 = \sum_{i=1}^{m} \left( N_i W_i \right) \cdot C
\]

PI-ACT\textsubscript{j} defined in Eq.(3.3) is employed at the \( j \)-th ingress router to adjust the marking threshold of the \( j \)-th unsatisfied AF adaptive aggregate, \( j=1 \ldots m \). \( m \) is the number of unsatisfied AF adaptive aggregates.

\[
k_{PI-ACT_j} \left( \frac{s}{s_{PI-ACT_j}} + 1 \right) \\
PI-ACT_j(s) = \frac{s_{PI-ACT_j}}{s} \\
(3.3)
\]

The aggregate arriving rate at the ingress router is computed by measuring the number of sent packets over a fixed time period \( T_{TSW} \) and further smoothed by a low-pass filter \( F \). The transfer function representing this estimation is given by

\[
F(s) = \frac{a}{s + a} e^{-sT_{TSW}} \\
(3.4)
\]

The transfer function representing RED mechanism for OUT traffic is given by

\[
AQM_j(s) = \frac{L_{RED}}{s} + 1 \\
(3.5)
\]

Figure 3.2 is the closed-loop block diagram of the model described by Eqs.(3.2)–(3.5).

### 3.2.3 Local stability analysis

In the following we first give the **Small Gain Theorem** and then apply it to analyze the system local stability.
Figure 3.2  The combined PI-ACT/RIO AF-based DiffServ network

**Small Gain Theorem [63]:** Consider the feedback system shown in Figure 3.3, where \( \hat{P} \) and \( \Delta \) are stable linear systems. If \( |\hat{P}\Delta| < 1 \) for all \( \omega \) then the feedback system is stable. Here \( \hat{P} \) denotes the nominal system; \( \Delta \) denotes the total perturbation.

The block diagram in Figure 3.2 is redrawn in Figure 3.4. We define the TCP/AQM network without the AF service as the nominal system. Thus,

\[
\hat{P}(s) \triangleq \frac{\delta p}{\delta \mathbf{x}} = \frac{1}{s \frac{\delta f}{\delta q}^{\text{AQM}} - 1} \frac{1}{s \frac{\delta f}{\delta q}} \left( \sum_{i=1}^{n} P_i \right)
\]  

(3.6)
Figure 3.3 Simple feedback system

Figure 3.4 Block diagram of AQM networks with active PI loops (the perturbation block indexes correspond to those in the set $J$)
where $\delta \tilde{x}$ is the total rate perturbation from the nominal TCP/AQM value due to active PI-ACTs:

$$\delta \tilde{x} = \sum_{i \in J} \delta \tilde{x}_i \quad \text{and} \quad \delta \tilde{x}_i = P_i \Delta_i \delta \rho$$

The TCP transfer function $P_i$ of the $i$-th aggregate is given by

$$P_i = e^{-sR_i} \frac{\partial g_i}{\partial p_i} \frac{1}{s - \frac{\partial g_i}{\partial W_i}} \frac{N_i}{R_i}$$

The perturbation $\Delta_j(s)$ induced by the $j$-th PI-ACT is described by

$$\Delta_j(s) = \frac{1}{s - \frac{\partial g_j}{\partial W_j}} \left( \frac{\partial g_j}{\partial CIR_j} \right) F_j(s)$$

Then the total perturbation is

$$\Delta(s) = \sum_{j \in J} P \Delta_j(s)$$

To derive the conditions to make the linearized DiffServ network, described by Eqs.(3.2)–(3.5), locally stable, it is sufficient to derive the conditions to guarantee (i) the stability of $\hat{P}$ and $\Delta$ and (ii) $\| \hat{P} \Delta \|_{\infty} = \sup_j \| \hat{P}_j \Delta_j \| < 1$.

### 3.2.3.1 The stability of $\Delta$

To analyze the stability of $\Delta$, it is sufficient to discuss the stability of $\Delta_j$ as follows. We use Nyquist stability criteria to show that there exists $k_{\text{PI-ACT}} > 0$ stabilizing $\Delta_j$. $\Delta_j$ can be written as
\[ \Delta_j(s) = \frac{L_{\Delta_j}(s)}{1 + L_{\Delta_j}(s)} \]

where

\[ L_{\Delta_j}(s) = \frac{1}{s} \frac{N_j}{R_j} \frac{\partial g_j}{\partial W_j} (\text{Pl-Act}_j) \frac{F_j(s)}{\partial \text{CIR}^\text{Th}_j} \]

\[ = \frac{1}{s} \frac{N_j}{R_j} \frac{\partial g_j}{\partial W_j} \left( \frac{k_{\text{Pl-Act}_j}}{z_{\text{Pl-Act}_j} + 1} \right) \frac{a_j e^{-r_{\text{TSW}}}}{s + a_j} \]

(3.7)

Since \( L_{\Delta_j}(s) \) has a pole at the origin, it is necessary that the Nyquist contour \( \Gamma \) includes an infinitesimal semicircle \( \Gamma_\varepsilon \) around \( s=0 \) described by

\[ \Gamma_\varepsilon = \left\{ s = \varepsilon e^{i\theta}; \; \theta \in \left[-90^\circ, 90^\circ\right], \varepsilon \to 0, \varepsilon > 0 \right\} \]

(3.8)

As \( s \) traverses from \(-\varepsilon i\) to \(+\varepsilon i\) along \( \Gamma_\varepsilon \), \( \theta \) changes from \(-90^\circ\) to \(+90^\circ\) counterclockwise. The corresponding Nyquist plot of can be determined by evaluating Eq.(3.7) along Eq. (3.8). In the limit we have

\[ \lim_{\varepsilon \to 0} L_{\Delta_j}(\varepsilon e^{i\theta}) = \frac{1}{s} \frac{N_j}{R_j} \frac{\partial g_j}{\partial W_j} \frac{k_{\text{Pl-Act}_j}}{\varepsilon e^{i\theta}} \]

The contour indentation near the origin \( \Gamma_\varepsilon \) is mapped by \( L_{\Delta_j}(s) \) into a semi-infinite circle covering the RHP of the complex plane. Any instability in \( \Delta_j(s) \) will be a result of encirclements by the Nyquist plot of \( L_{\Delta_j}(s) \) over the range \( \omega \in (\varepsilon, +\infty) \cup (-\varepsilon, -\infty) \). Define \( L_{\Delta_j}(j\omega) = k_{\text{Pl-Act}_j} \tilde{L}_{\Delta_j}(j\omega) \). The plot of \( \tilde{L}_{\Delta_j}(j\omega) \) crosses the negative real-axis at frequencies in the set \( \Omega = \left\{ \omega: \angle \tilde{L}_{\Delta_j}(j\omega) = -180^\circ \right\} \). Let \( \omega_1 \) be the frequency such
that \( |\vec{L}_{\Delta_j}(j\omega)| = \max_{\omega \in \Omega} |\vec{L}_{\Delta_j}(j\omega)| \). If \( k_{\text{PL-ACT}_j} < \left| \vec{L}_{\Delta_j}(j\omega) \right|^{-1} \) then \( \left| \vec{L}_{\Delta_j}(j\omega) \right| < 1 \), implying stability of \( \Delta_j \). Stability of \( \Delta \) follows immediately from

\[
k_{\text{PL-ACT}_j} < \frac{1}{\left| \vec{L}_{\Delta_j}(j\omega) \right|}, \quad j = 1, \ldots, n
\]

### 3.2.3.2 The conditions to stabilize \( \hat{P} \)

Now we use Nyquist stability argument to derive an upper bound on the AQM gain, sufficient to stabilize \( \hat{P} \). We define

\[
\hat{P}(s) = \frac{1}{s - \frac{\partial f}{\partial q} \frac{s}{k_{\text{RED}}} + 1} \frac{L_{\text{RED}}}{1 + L(s)}
\]

and then we obtain

\[
L(s) = \frac{1}{s - \frac{\partial f}{\partial q} \frac{s}{k_{\text{RED}}} + 1} \sum_{i=1}^{n} e^{-\alpha_i} \frac{\partial g_i}{\partial x} s - \frac{\partial g_i}{\partial W_i} \frac{1}{R_i} N_i
\]

Any instability in \( \hat{P}(s) \) will be the result of encirclements by the Nyquist plot of \( L(s) \) over the range \( \omega \in (-\infty, +\infty) \). Define \( L_{\hat{P}}(j\omega) = L_{\text{RED}} \times \hat{L}_{\hat{P}}(j\omega) \). The plot of \( \hat{L}_{\hat{P}}(j\omega) \) crosses the negative real-axis at frequencies in the set \( \Omega = \{ \omega : \angle \hat{L}_{\hat{P}}(j\omega) = -180^\circ \} \). Let \( \omega_2 \) be the frequency such that \( |\hat{L}_{\hat{P}}(j\omega_2)| = \max_{\omega \in \Omega} |\hat{L}_{\hat{P}}(j\omega)| \). If

\[
L_{\text{RED}} < \frac{1}{|\hat{L}_{\hat{P}}(j\omega_2)|}
\]

then \( |\hat{L}_{\hat{P}}(j\omega)| < 1 \) implying the stability of \( \hat{P} \).
3.2.3.3 Conditions for $|\hat{P}(j\omega)\Delta(j\omega)| < 1$

Now we show that $|\hat{P}(j\omega)\Delta(j\omega)| < 1$ for all $\omega$. We first show that $|\Delta_j(j\omega)| = 1$ which is used later.

1. conditions for $|\Delta_j(j\omega)| = 1$

Here $\text{Re}\left(\hat{L}_\Delta(j\omega)\right)$ denotes the real part of $\hat{L}_\Delta(j\omega)$. Because $L_{\Delta_j}(j\omega) = k_{\text{PL-ACT}_j} \hat{L}_\Delta(j\omega)$, then $\text{Re}\left(\hat{L}_\Delta(j\omega)\right) = k_{\text{PL-ACT}_j} \text{Re}\left(\hat{L}_\Delta(j\omega)\right)$. It is obvious that $\left|\frac{L_{\Delta_j}(s)}{1 + L_{\Delta_j}(s)}\right| < 1$ as long as $k_{\text{PL-ACT}_j} \text{Re}\left(\hat{L}_\Delta(j\omega)\right) > -\frac{1}{2}$.

Now we derive the conditions for $k_{\text{PL-ACT}_j} \text{Re}\left(\hat{L}_\Delta(j\omega)\right) > -\frac{1}{2}$. When $\text{Re}\left(\hat{L}_\Delta(j\omega)\right) > 0$, it is true that $k_{\text{PL-ACT}_j} \text{Re}\left(\hat{L}_\Delta(j\omega)\right) > -\frac{1}{2}$ because $k_{\text{PL-ACT}_j} > 0$. When $\text{Re}\left(\hat{L}_\Delta(j\omega)\right) < 0$, $k_{\text{PL-ACT}_j} \text{Re}\left(\hat{L}_\Delta(j\omega)\right) > -\frac{1}{2}$ if $k_{\text{PL-ACT}_j} < \frac{1}{2\text{Re}\left(\hat{L}_\Delta(j\omega)\right)}$. Then we look for the smallest $\frac{1}{2\text{Re}\left(\hat{L}_\Delta(j\omega)\right)}$, that is, the smallest $\text{Re}\left(\hat{L}_\Delta(j\omega)\right)$. Let $\omega_3$ be the frequency where $\text{Re}\left(\hat{L}_\Delta(j\omega)\right) = \min_{\omega \in \Gamma} \text{Re}\left(\hat{L}_\Delta(j\omega)\right)$.

Thus, when $0 < k_{\text{PL-ACT}_j} < \frac{1}{2\text{Re}\left(\hat{L}_\Delta(j\omega)\right)}$, $\text{Re}\left(\hat{L}_\Delta(j\omega)\right) > -\frac{1}{2}$. That is, $|\Delta_j(j\omega)| = \left|\frac{L_{\Delta_j}(s)}{1 + L_{\Delta_j}(s)}\right| < 1$ for all $j$. Because a pole of $L_{\Delta_j}(j\omega)$ is zero, then $|\Delta_j(j\omega)| = 1$ at $\omega = 0$. (It is due to when $\omega$ is 0, PI-ACT is going towards to infinity.) Thus,
\[ \| \Delta_j(j\omega) \|_\infty = 1 \Rightarrow k_{PL,ACT,j} < \min \left\{ \frac{1}{2 \text{Re} \left( \tilde{\Delta}_j(j\omega) \right)}, \frac{1}{\text{Re} \left( \tilde{\Delta}_j(j\omega) \right)} \right\} \]

2. Condition for \( |\hat{P}(j\omega)\Delta(j\omega)| < 1 \)

Let \( L(s) = -\frac{1}{s - \frac{\partial f}{\partial q}} \sum_{i=1}^{N} e^{-j\omega_k} \frac{\partial G_i}{\partial p} s - \frac{\partial G_i}{\partial W_i} R_i \). We obtain

\[ |L(j\omega)| = \left| \frac{1}{j\omega - \frac{\partial f}{\partial q}} \sum_{i=1}^{N} e^{-j\omega_k} \frac{\partial G_i}{\partial p} \frac{1}{j\omega - \frac{\partial G_i}{\partial W_i}} R_i \right| \]

\[ |L(j\omega)| \leq \left| \frac{1}{j\omega - \frac{\partial f}{\partial q}} \sum_{i=1}^{N} \frac{\partial G_i}{\partial p} \frac{1}{j\omega - \frac{\partial G_i}{\partial W_i}} R_i \right| \]

\[ \therefore \frac{\partial G_i}{\partial W_i} < 0 \text{ and } \frac{\partial f}{\partial q} < 0 \]

\[ \therefore |L(j\omega)| \leq \left| \frac{1}{\partial q} \sum_{i=1}^{N} \frac{\partial G_i}{\partial p} \left| \left| \frac{1}{\partial W_i} \right| R_i \right| \right| \]

Let \( M = \left| \sum_{i=1}^{N} \frac{\partial G_i}{\partial p} \frac{1}{\partial W_i} \right| \). Thus \( |L(j\omega)| < \varepsilon_1 \) where \( \varepsilon_1 \) is in \((0,1,0)\) as long as

\[ L_{RED} < \frac{\varepsilon_1}{M} \quad (3.9) \]
Here we choose \( \varepsilon_1 \) in \((0, 1.0)\) in order to \(\left| \frac{1}{1 - |L(j\omega)|} \right| < \left| \frac{1}{1 - |L(j\omega)|} \right|_\infty\).

Thus

\[
|\hat{P}(j\omega)| = \left| \frac{1}{j\omega - \frac{\partial f}{\partial q} \text{AQM}} \right| < \left| \frac{1}{j\omega - \frac{\partial f}{\partial q} \text{AQM} \left( \sum_{i=1}^n R_i \right)} \right| < \left| \frac{1}{j\omega - \frac{\partial f}{\partial q} \text{AQM} \left( \sum_{i=1}^n R_i \right)} \right| = \left| \frac{L_{\text{RED}}}{j\omega - \frac{\partial f}{\partial q} \left( \frac{j\omega}{k_{\text{RED}}} + 1 \right)} \right|
\]

Now we bound \( \Delta \) using

\[
|\Delta(j\omega)| = \left| \sum_{j \in J} P_j(j\omega) \Delta_j(j\omega) \right| < \sum_{j \in J} |P_j(j\omega)\Delta_j(j\omega)| < \sum_{j \in J} |P_j(j\omega)| \left| \Delta_j(j\omega) \right| \leq \sum_{j \in J} |P_j(j\omega)|
\]

Assuming \( k_{\text{max}} = \max_{j \in J} \left\{ \frac{\partial g_j}{\partial p} \frac{N_j}{R_j} \right\} \) and \( p_{\text{min}} = \min_{j \in J} \left\{ -\frac{\partial g_j}{\partial W_j} \right\} \), we obtain

\[
|\hat{P}(j\omega)\Delta(j\omega)| < |\hat{P}(j\omega)| \left| \sum_{j \in J} \left( |P_j(j\omega)| \left| \Delta_j(j\omega) \right| \right) \right| \leq |\hat{P}(j\omega)| \sum_{j \in J} |P_j(j\omega)|
\]

\[
|\hat{P}(j\omega)\Delta(j\omega)| < \left| \frac{1}{j\omega - \frac{\partial f}{\partial q} \frac{j\omega}{k_{\text{RED}}} + 1} \right| \times \sum_{j \in J} e^{-j\omega t} \frac{\partial g_j}{\partial p} \frac{1}{j\omega - \frac{\partial g_j}{\partial W_j} \frac{N_j}{R_j}}
\]

\[
|\hat{P}(j\omega)\Delta(j\omega)| < \left| \frac{1}{\frac{\partial f}{\partial q} \left( \frac{1}{1 - \varepsilon_1} \right)} \right| \times \sum_{j \in J} \left( \frac{\partial g_j}{\partial p} \left| \frac{1}{\frac{\partial g_j}{\partial W_j} \frac{N_j}{R_j}} \right| \right)
\]

\[
|\hat{P}(j\omega)\Delta(j\omega)| < \left| \frac{L_{\text{RED}}}{\frac{\partial f}{\partial q} \left( \frac{1}{1 - \varepsilon_1} \right)} \right| \times \frac{k_{\text{max}}}{p_{\text{min}}}
\]

Define

35
\[
L_{\text{RED}} < \min \left\{ \frac{m}{k_{\text{max}}} \frac{\varepsilon_i}{P_{\text{min}}} \left| \frac{\partial f}{\partial q} \left( 1 - \varepsilon_i \right) \right| \frac{1}{M}, \frac{\varepsilon_i}{M} \right\} \Rightarrow \left| \hat{P}(j\omega) \Delta(j\omega) \right| < 1
\]

Finally, the AF-based DiffServ network described by Eqs.(3.2)-(3.5) is locally stable if \( L_{\text{RED}} \) and \( k_{\text{PI-\text{ACT}}} \) satisfy

\[
\begin{align*}
   k_{\text{PI-\text{ACT}}} &< \min \left\{ \frac{1}{2 \left| \text{Re} \left( \hat{L}_{\beta}(j\omega_j) \right) \right| \left| \hat{L}_{\alpha}(j\omega_j) \right|} \right\}, \quad j = 1, \ldots, m \\
   L_{\text{RED}} &< \min \left\{ \frac{\varepsilon_i}{M}, \frac{1}{M}, \frac{1}{\left| \hat{L}_{\beta}(j\omega_2) \right|} \right\}
\end{align*}
\]

(3.10)

### 3.2.4 An illustrative example

In this subsection, we apply the above sufficient conditions to analyze stability of a simple under-subscribed AF-based DiffServ network. This network consists of three heterogeneous aggregates (A1-A3). A1-A3 consist of 20, 30 and 25 micro-flows, respectively. All the micro-flows (FTP flows) in an aggregate have the same characteristics. The round-trip link delays of A1, A2 and A3 are set to 0.23 second, 0.1 second and 0.05 second, respectively. CIR₁=2000 packets, CIR₂=500 packets, CIR₃=1250 packets. Core router buffer size is set to 1200 packets. Link capacity is 4500 packets. Thus only PI-\( \text{ACT}_1 \) is active.

We set \([q_{\text{min}}, q_{\text{max}}, p_{\text{max}}, q_{\text{weight}}]\) for OUT packets to \([600 \text{ packets}, 50 \text{ packets}, 0.25, 0.0000011111]\). Thus the AQM controller used for OUT traffic at the core router is

\[
AQM_1(s) = \frac{4.5455 \times 10^{-4}}{s + 0.005} + 1
\]

We choose

\[
\text{PI-\text{ACT}}_1(s) = \frac{s}{0.6} \quad \text{and} \quad F(s) = \frac{1}{s+1} e^{-s}
\]
That is, the arriving rate of an aggregate at the ingress node is estimated per one second. In PI-ACT, the proportional gain is 0.001 and the integral gain is 0.0006.

Queue length oscillates around 100 packets. Hence, the round trip times are $R_1 = 0.2522$ s, $R_2 = 0.1222$ s, $R_3 = 0.0722$ s. The nominal TCP/AQM system is described by

$$|\hat{p}(j\omega)| = \left| \frac{1}{j\omega - \frac{df}{dq}} \frac{AQM}{1 - \frac{1}{j\omega - \frac{df}{dq}} AQM \left(\sum_{i=1}^{3} P_i\right)} \right|$$

where $\frac{df}{dq} = -8.1930$ 1/second; and where the transfer functions for $A_1$, $A_2$ and $A_3$ are described by

$$P_1 = e^{-sR_1} \frac{\partial g_1}{\partial p}, \quad \frac{1}{s - \frac{\partial g_1}{\partial W}} \frac{N_1}{R_1} = e^{-s(0.2522)} (-223.0209) \frac{1}{s - (-0.9733)} \frac{20}{0.2522}$$

$$P_2 = e^{-sR_2} \frac{\partial g_2}{\partial p}, \quad \frac{1}{s - \frac{\partial g_2}{\partial W}} \frac{N_2}{R_2} = e^{-s(0.1222)} (-73.6498) \frac{1}{s - (-1.3460)} \frac{30}{0.1222}$$

$$P_3 = e^{-sR_3} \frac{\partial g_3}{\partial p}, \quad \frac{1}{s - \frac{\partial g_3}{\partial W}} \frac{N_3}{R_3} = e^{-s(0.0722)} (-186.9806) \frac{1}{s - (-2.8477)} \frac{25}{0.0722}$$

The disturbance is described by $\Delta_t(s) = \frac{L_{\Delta_t}(s)}{1 + L_{\Delta_t}(s)}$, where

$$L_{\Delta_t}(s) = \frac{1}{s - \frac{\partial g_1}{\partial W_1}} F_t(s) = e^{-s}$$

$$= \frac{1}{s - (-0.9733)} \frac{20}{0.2522} (0.0186) \frac{6 \times 10^{-4} \left(\frac{s}{0.6} + 1\right)}{s} \frac{1}{s + 1} e^{-s}$$

We observe that in Figure 3.5 $|\hat{p}(\mu)\Delta_t| < 1$, in Figure 3.6 $|\Delta_t| < 1$, and in Figure 3.7. $|\hat{p}| < 1$, which establish local stability of the example network. In addition, we give the $n_s$-2 simulation results. Figure 3.8 shows the Average Goodput variation and $CIR_{\text{thresh}}$. 

37
Figure 3.5  Magnitude Bode plot of $(\hat{p} \Delta)$  Figure 3.6  Magnitude Bode plot of $\Delta$

Figure 3.7  Magnitude Bode plot of $\hat{p}$

Figure 3.8  Goodput and $CIR_{\text{Thresh}}$ variations of $A_1$-$A_3$

variations of $A_1$-$A_3$. Figure 3.9 gives the bottleneck link queue length variation versus time.

3.2.5  Relations between network parameters and $k_{\text{PL-AC1..j}}$

We now analyze the relations between network parameters $(N_j$ and $R_j)$ and $|\tilde{L}_{n_j}(s)|$, and the relations between network parameters $(N_j$ and $R_j)$ and $|\text{Re}\left(\tilde{L}_{n_j}(s)\right)|$. When
\[ \frac{1}{L_{\alpha_j}(j\omega)} = \left| W_j - \left( \frac{R CIR_j^{th}}{N_j} \left( 1 - \frac{1}{2 W_j^2} \right) \right) \right| \left| \frac{j\omega_j}{W_j} \right| \frac{1}{a_j} \frac{1}{e^{-sT_{SW_j}}} \]

and

\[ \left| \text{Re}(\tilde{L}_{\alpha_j}(s)) \right| = \left| \frac{1}{W_j - \frac{R CIR_j^{th}}{N_j} \left( 1 - \frac{1}{2 W_j^2} \right)} \left( \frac{1}{W_j} + \frac{W_j}{2} \right) \frac{1}{1 + \omega_i^2} \left( \cos(T_{SW_j}) \frac{1}{a_j} - \omega \sin(T_{SW_j}) \right) \right| \]

Thus, the upper bound of \( \tilde{L}_{\alpha_j}(s) \) is decreasing function of \( CIR_j^{th} \) and \( R_i \), increasing function of \( N_i \). Same relations are for \( \left| \text{Re}(\tilde{L}_{\alpha_j}(s)) \right| \). Thus, \( K_p \) and \( K_i \) are both increasing function of \( CIR_j^{th} \) and \( R_i \), decreasing function of \( N_i \).

### 3.3 Analysis of some existing mechanisms

In this section, we first give the tuning rules of a generic NPI-type controller structure. Then, we use it to analyze several existing adaptive mechanisms, including PME [25], MBM [43], ARM [14], and ACT [12].
3.3.1 The rules of tuning controller gains

Eq.(3.11) gives the digital approximate implementation of Eq.(3.1), obtained by applying Trapezoidal rule [20].

\[
CIR_{\text{thresh}}(k) = CIR_{\text{thresh}}(k - 1) + K_p (e(k) - e(k - 1)) + K_i \left( e(k) + e(k - 1) \right) \tag{3.11}
\]

Eq.(3.11) is the combination of the proportional control action \( CIR_{\text{pro}}(k) = K_p e(k) \) and the integral control action \( CIR_{\text{int}}(k) = CIR_{\text{int}}(k - 1) + K_i (e(k) + e(k - 1)) \). Although a fixed-gain PI controller is an effective controller for a static system, its ability is degraded in the presence of network parameter variations and disturbances. A controller with varying gains, tuned appropriately, can produce better performance than the fixed-gain controller, while still enjoying the simple structure of the fixed-gain PI controller.

Designing an efficient NPI controller requires proper tuning of \( K_p \) and \( K_i \) in order to produce small overshoot, short settling time, small steady state error and low sensitivity in dynamic networks. Settling time reflects how fast the bandwidth assurance is re-achieved after changes in the network conditions. Overshoot represents the value of \( \frac{r_s^{\text{max}}}{CIR} \), where \( r_s^{\text{max}} \) is the maximum value of \( r_s \) during its transient phase. A large overshoot damages the benefits of other aggregates. Sensitivity represents how significantly the changes in network resources or traffic characteristics affect the attainment of bandwidth assurance for unsatisfied aggregates. It describes the robustness of the control system with respect to these changes.

As we have discussed, \( K_p \) and \( K_i \) can be tuned by applying the indirect adaptive control approach. In this chapter we attempt the system-state-based self-tuning method to adjust \( K_p \) and \( K_i \). In this method, \( K_p \) and \( K_i \) are designed as functions of the system output, \( r_s \).

Now we discuss how to use \( |e(k)| \) to adjust \( K_p \) and \( K_i \) from two standpoints.

(i) Relations between network parameters and \( k_{P,ACT,j} \)

When an unsatisfied aggregate consists of a larger number of micro-flows, its ability in grabbing bandwidth must be strengthened and then \( |e(k)| \) is small. That is, a small
increase in $CIR_{\text{Thresh}}$ can achieve bandwidth assurance. The conclusions in Section 3.2.5 show that $K_p$ and $K_i$ are decreasing function of $N_i$. Thus, $K_p$ and $K_i$ are increasing functions of $|e(k)|$.

When the RTT of the micro-flows of an aggregate, its ability in grabbing bandwidth must be weakened and then $|e(k)|$ is large. That is, a larger increase in $CIR_{\text{Thresh}}$ can achieve bandwidth assurance. The conclusions in Section 3.2.5 show that $K_p$ and $K_i$ are both increasing function of $R_i$. Thus, $K_p$ and $K_i$ are increasing functions of $|e(k)|$.

When CIR of an aggregate is larger, its ability in grabbing bandwidth must be strengthened in order to reach CIR and then $|e(k)|$ is large. That is, a larger increase in $CIR_{\text{Thresh}}$ can achieve bandwidth assurance. The conclusions in Section 3.2.5 show that $K_p$ and $K_i$ are both increasing function of $CIR_{\text{Thresh}}$. Thus, $K_p$ and $K_i$ are increasing functions of $|e(k)|$.

(ii) The features of the proportional control action and the integral control action.

When the characteristics of a satisfied aggregate or the environmental conditions (such as characteristics of other aggregates) change, the aggregate’s ability in grabbing bandwidth may change. That is, the operating point of $CIR_{\text{Thresh}}$ of this aggregate may change. This change must result in changes in $|e(k)|$. Usually, the larger the $|e(k)|$, the larger the impact of the network conditions on the aggregate. That is, a large $|e(k)|$ usually means that the current $CIR_{\text{Thresh}}$ is far away from the new operating point. The proportional control action, $K_p(k)e(k)$, changes $CIR_{\text{Thresh}}$ in proportion to the value of $e(k)$ and in the direction, which reduces $e(k)$ [3]. The integral action changes $CIR_{\text{Thresh}}$ incrementally, in proportion to the time integral of previous errors.

A large $K_p$ and $K_i$ can produce faster transient response with possible instability [3]. Thus, when there is no disturbance, it is reasonable to design $K_p$ and $K_i$ as increasing functions of $|e(k)|$. Then $CIR_{\text{Thresh}}$ can quickly reach the new operating point. In addition, the possible instability caused by using large constant $K_p$ and $K_i$ to speed up transient response is alleviated. However, when considering disturbance, the rules of adjusting $K_i$ are becoming complex. Thus, in Section 3.3.2 we only analyze the proportional gain in each mechanism. As shown in the simulation results in Section 3.5, this analysis method can provide insights into the behaviors of PME, MBM, ARM, and ACT.
3.3.2 Analysis of some existing mechanisms

We use the above discussions to analyze PME, MBM, ARM and ACT. We have mentioned that $r_a$ is a low-pass filtered average arriving rate. A fluid model for this dynamics is given in Eq.(3.4). In order to simplify the analysis, we ignore this dynamic. In addition, we ignore the low-pass filter when we analyze ARM. Based on these assumptions, we can map PME, MBM, ARM, and ACT to the NPI-type controller structure in Eq.(3.11).

3.3.2.1 PME

PME uses Eq.(3.12) to update $mp(k)$ at time instant $k$. $\eta$ is a positive constant.

$$mp(k) = mp(k-1) + \eta \left( 1 - \frac{r_a(k)}{CIR} \right)$$  \hspace{1cm} (3.12)

By letting $e(k)$=CIR- $r_a(k)$ and $CIR_{Thresh}=mp(k)$ $r_a(k)$, we obtain

$$CIR_{Thresh}(k) = CIR_{Thresh}(k-1) + \frac{r_a(k)}{CIR} \eta \left( e(k) - e(k-1) \right) + \frac{r_a(k)}{CIR} \eta \left( e(k) + e(k-1) \right)$$  \hspace{1cm} (3.13)

Thus, $K_p(k) = \frac{r_a(k)}{CIR} \eta$, showing that: (i) When $r_a$<CIR, the controller is a NPI controller, where $K_p$ is a non-decreasing function of $r_a$ and a non-increasing function of CIR. Thus, an unsatisfied aggregate with small $r_a$ or with large CIR has a slow speed of increasing $CIR_{Thresh}$. (ii) When $r_a$>CIR, the controller is also a NPI controller. It is obvious that $K_p$ in the case of $r_a$<CIR is always smaller than $K_p$ in the case of $r_a$>CIR. When an aggregate exceeds its CIR through increasing $CIR_{Thresh}$, the decreasing action of $K_p$ causes the arriving rate to quickly go below CIR. However because the speed of increasing $CIR_{Thresh}$ is slow, it takes a long time to recover back towards the CIR from below. Thus, the average goodput is below CIR for a long time. Thus, it is difficult for the unsatisfied aggregates to approximate the average goodput close to their CIRs. Note that such decreasing and increasing methods result in the small difference in the achieved average goodput among the AF adaptive aggregates, compare to TSW.
3.3.2.2 Memory-based marker

MBM uses Eq.(3.14) to update mp. \( \nu(k) \) denotes the average arriving rate estimated at time instant \( k \).

\[
\begin{align*}
mp(k) &= mp(k - 1) + \left( 1 - \frac{\nu(k)}{CIR} \right) + \frac{\nu(k) - \nu(k - 1)}{\nu(k)} \quad \nu(k) \leq CIR \\
mp(k) &= mp(k - 1) + \frac{\nu(k - 1) - \nu(k)}{\nu(k)} \quad \nu(k) > CIR
\end{align*}
\] (3.14)

We rewrite Eq.(3.14) into Eq.(3.15) by letting \( e(k) = CIR - \nu(k) \) and \( CIR_{\text{thresh}} = \nu(k)mp(k) \).

\[
\begin{align*}
\text{if} \quad \nu(k) \leq CIR & \\
CIR_{\text{thresh}}(k) &= CIR_{\text{thresh}}(k - 1) + \left( \frac{1}{2} \frac{\nu(k)}{CIR} + 1 \right)(e(k) - e(k - 1)) + \frac{1}{2} \frac{\nu(k)}{CIR}(e(k) + e(k - 1)) \\
\text{if} \quad \nu(k) > CIR & \\
CIR_{\text{thresh}}(k) &= CIR_{\text{thresh}}(k - 1) + (e(k) - e(k - 1))
\end{align*}
\] (3.15)

Note that MBM updates \( mp \) whenever a packet arrives. Eq.(3.15) shows that (i) When \( \nu(k) \leq CIR \), \( K_s(k) = \frac{\nu(k)}{2 \cdot CIR} + 1 \). Thus, the controller is an NPI-type controller. The features of the MBM in this case are similar to PME. The unsatisfied aggregate with smaller \( r_s \) or with larger CIR has a slow speed of increasing \( CIR_{\text{thresh}} \). (ii) When \( \nu(k) > CIR \), the controller is a fixed-gain Proportional (P) controller. The static feature when \( \nu(k) > CIR \) and the slow increasing feature when \( \nu(k) \leq CIR \) enlarges the difference in the achieved Average Goodput among the competing AF adaptive aggregates, compared to TSW.

3.3.3 ARM

ARM consists of a fixed-gain PI controller and a low-pass filter. We ignore the low-pass filter. Then it can be mapped to Eq.(3.1). The token bucket rate is a kind of \( CIR_{\text{thresh}} \). \( a \) and \( b \) are positive constants.

\[
CIR_{\text{thresh}}(k) = CIR_{\text{thresh}}(k - 1) + ae(k) - b(k - 1)
\] (3.16)

Eq.(3.16) is a fixed-gain PI controller, where \( K_c(k) = \frac{a + b}{2} \) and \( K_s(k) = \frac{a - b}{2} \). We have previously mentioned the disadvantage of a fixed-gain controller. But the following simulation results show that ARM produces a fast response in most cases. The main
reason is that ARM uses zero as the lower bound of $CIR_{\text{Threshold}}$. We discuss the drawbacks of using zero as the lower bound in Section 3.4.2.2.

### 3.3.3.1 ACT marker

ACT uses Eq.(3.17) to update $CIR_{\text{Threshold}}$ at time instant $k$. $\gamma$ and $\beta$ are both positive constants.

\[
\begin{cases}
    CIR_{\text{threshold}}(k) = CIR_{\text{threshold}}(k-1) + \gamma \text{CIR} \left( 1 - \frac{r_s(k)}{CIR} \right) & r_s(k) \leq \text{CIR \& \& CIR}_{\text{threshold}} < 2.0 \text{CIR} \\
    CIR_{\text{threshold}}(k) = CIR_{\text{threshold}}(k-1) - \beta \text{CIR} \left( 1 - \frac{CIR}{r_s(k)} \right) & r_s(k) > \text{CIR \& \& CIR}_{\text{threshold}} > \text{CIR}
\end{cases}
\]  

(3.17)

We rewrite Eq.(3.17) into Eq.(3.18) by letting $e(k) = \text{CIR} - r_a(k)$.

\[
\begin{cases}
    \text{if } r_s(k) \leq \text{CIR \& \& CIR}_{\text{threshold}} < 2\text{CIR} \\
    CIR_{\text{threshold}}(k) = CIR_{\text{threshold}}(k-1) + \frac{\gamma}{2} (e(k) - e(k-1)) + \frac{\beta}{2} (e(k) + e(k-1)) \\
    \text{if } r_s(k) > \text{CIR \& \& CIR}_{\text{threshold}} > \text{CIR} \\
    CIR_{\text{threshold}}(k) = CIR_{\text{threshold}}(k-1) + \frac{1}{2} \frac{\beta \text{CIR}}{r_s(k)} (e(k) - e(k-1)) + \frac{1}{2} \frac{\beta \text{CIR}}{r_s(k)} (e(k) + e(k-1))
\end{cases}
\]  

(3.18)

Eq.(3.18) shows that (i) When $r_s \leq \text{CIR}$, the controller is a fixed-gain PI controller, where $K_p(k) = \frac{\gamma}{2}$ and $K_i(k) = \frac{\beta}{2}$. (ii) When $r_s > \text{CIR}$, the controller is a NPI controller, where $K_p(k) = \frac{1}{2} \frac{\beta \text{CIR}}{r_s(k)}$, a non-increasing function of $r_s$ and non-decreasing function of CIR. Thus, $K_p(k)$ is a non-increasing function of $|e(k)|$. It is noted that the features of the control gains in ACT in this case are opposite to those of PME. This slowly-decreasing feature in adjusting the $CIR_{\text{Threshold}}$ of the conditionally-satisfied aggregate with large $r_s$ when $r_s > \text{CIR}$ is undesirable. In addition, setting $\gamma$ larger than $\beta$ in [12] also reduces the speed of decreasing $CIR_{\text{Threshold}}$. In the following sections we use the term slowly-decreasing method to represent the use of these two features that cause the slow speed in decreasing $CIR_{\text{Threshold}}$ of the conditionally-satisfied aggregate. Although the slowly-decreasing method can accelerate the attainment of bandwidth assurance for some unsatisfied aggregates, it may result in an excessive increase in $CIR_{\text{Threshold}}$ of some conditionally-satisfied aggregates.
This results in the Average Goodput of these aggregates being larger than their CIRs. A serious side effect is that the excessive amount of IN traffic may prevent weak unsatisfied aggregates, such as those with large RTT, large CIR and the like, from improving their goodput. It may also result in the unfair sharing of excess bandwidth among aggregates. We see these effects in the simulations presented later.

3.3.3.2 Summary

According to the above analysis, we list $K_p$ and $K_i$ of each mechanism in Table I. From this table, we can see that either these mechanisms are fixed-gain controllers or the controller gains are adjusted contrary to what is desired. As shown in the following simulation results, a fixed-gain controller performs well in some network situations but performs worse in other situations. However, a controller with an undesirable design of controller gains either results in the bandwidth attainment over CIR or can’t improve bandwidth assurance.

3.4 Variable-structure PI controller for adapting CIR threshold

In this section we present VS-ACT and discuss some design considerations.

3.4.1 The VS-ACT mechanism

Based on the above discussions, we develop a Variable-Structure PI controller for adapting $CIR_{\text{Thresh}}$. The initial value of $CIR_{\text{Thresh}}$ is set to CIR. The VS-ACT mechanism acts as follows: (i) when \( r_s < \text{CIR} \), $CIR_{\text{Thresh}}$ is increased step by step until $2\text{CIR}$; (ii) when \( r_s > \text{CIR} \) and $CIR_{\text{Thresh}} < \text{CIR}$, $CIR_{\text{Thresh}}$ is decreased step by step until CIR. The formula of adjusting $CIR_{\text{Thresh}}$ is depicted by

\[
\begin{align*}
CIR_{\text{Thresh}}(k) &= CIR_{\text{Thresh}}(k-1) + \phi(k) \left[ k_{\text{gain}} \left( e(k) - e(k-1) \right) + k_{\text{int}} \left( e(k) + e(k-1) \right) \right] \\
CIR_{\text{Thresh}}(k) &= \max \{ \text{CIR}, \min \{ 2\text{CIR}, CIR_{\text{Thresh}}(k) \} \}
\end{align*}
\]  

(3.19)

where
Table I Control gains in each scheme

<table>
<thead>
<tr>
<th></th>
<th>PME</th>
<th>MBM</th>
<th>ACT</th>
<th>ARM</th>
</tr>
</thead>
<tbody>
<tr>
<td>For unsatisfied aggregate</td>
<td>$K_p \frac{r_c(k) \eta}{CIR}$</td>
<td>$1 + \frac{1}{2 CIR}$ x</td>
<td>$\frac{\gamma}{2}$</td>
<td>$a + b$ fixed-gain</td>
</tr>
<tr>
<td>$K_i \frac{r_c(k) \eta}{CIR}$</td>
<td>$1 + \frac{1}{2 CIR}$</td>
<td></td>
<td>$\frac{\gamma}{2}$</td>
<td>$\frac{a - b}{2}$ fixed-gain</td>
</tr>
<tr>
<td>For satisfied aggregate</td>
<td>$K_p \frac{r_c(k) \eta}{CIR}$</td>
<td>$\sqrt{1}$ fixed-gain</td>
<td>$1 + \frac{1}{2 r_c(k)}$</td>
<td>$a + b$ fixed-gain</td>
</tr>
<tr>
<td>$K_i \frac{r_c(k) \eta}{CIR}$</td>
<td>$0$</td>
<td></td>
<td>$1 + \frac{1}{2 r_c(k)}$</td>
<td>$\frac{a - b}{2}$ fixed-gain</td>
</tr>
</tbody>
</table>

Note: x represents that the settings of controller gains have adverse effect on performance; $\sqrt{}$ represents that the settings of controller gains can bring help to the performance improvement; Fixed-gain represents that the controller gains are static.

$$
\varphi(k) = \beta \frac{k_{\text{max}}}{1 + \exp\left(-\left[\frac{e(k)}{\eta}\right]^2\right)} \\
\text{and} \quad \beta = \begin{cases} 
0.75 & |e(k)| < |e(k-1)| \\
1.0 & \text{otherwise}
\end{cases}
$$

$k_{\text{pmin}}$ and $k_{\text{imin}}$ in Eq.(3.19) and $k_{\text{max}}$ in Eq.(3.20) are positive constants. Their settings are discussed in Section 3.4.2.3. $\eta$ and $\beta$ are user-defined positive constants.

### 3.4.2 Design considerations

#### 3.4.2.1 The formula

Figure 3.10 depicts the block diagram of an AF-feedback-loop system with VS-ACT. It is easy to see that VS-ACT is based on modulating the control output of a fixed-gain PI controller with $\varphi(k)$, which is a modified sigmoidal function of $|e(k)|$. The reason for using the modified sigmoidal function rather than other kinds of functions such as the hyperbolic function or the piecewise-linear function is the consideration that (i) the exponential term can produce a fast increase or a fast decrease; (ii) it is much easier to bound the function value when using a smooth sigmoidal function; (iii) we need $K_p(k_1)=K_p(k_2)$ when $|e(k_1)|=|e(k_2)|$ when the moving directions at both time $k_1$ and time $k_2$
are the same (towards the CIR or away from CIR); thus we make modification to the standard sigmoidal function.

Eq.(3.19) and Eq.(3.20) show that $K_p(k)$ and $K_d(k)$ are both designed as non-decreasing functions of $|e(k)|$. The motivation of varying $K_i(k)$ proportional to $|e(k)|$ is that varying $K_i(k)$ proportional to $|e(k)|$ can produce fast transient response. In addition, the low-filtered arriving rate can accommodate some disturbance. Even if there is unnecessary accumulation in $CIR_{Thresh}$, the accumulation may be not large and may be quickly released because $K_i(k)$ is proportional to $|e(k)|$. In order to reduce the unnecessary accumulation due to disturbance and in order to prevent instability, we use small $k_{pmin}$ and $k_{imin}$ when $|e(k)|$ is small and the increasing speed of $K_p(k)$ and $K_i(k)$ is also small when $|e(k)|$ is not large. These are achieved by using $\eta$ and “square” in Eq.(3.20). Note that “square” can speed up the increase of $K_p(k)$ and $K_i(k)$ when $|e(k)| > \eta$. The time delay and the existence of the other aggregates may result in the excessive increase in $K_p(k)$ and $K_i(k)$, leading to instability. In order to reduce the possibly excessive increase/decrease in $CIR_{Thresh}$, $K_p(k)$ and $K_i(k)$ are varying between $[k_{pmin}, k_{max} + k_{pmin}]$ and $[k_{imin}, k_{max} + k_{imin}]$, respectively.
Now, we give the reason for using $\beta$. The control action of VS-ACT depends on the movement of $r_a$ toward CIR or away from CIR. Figure 3.11 is an example about the variation of $e(k)$ over time. The controller gains should be smaller in thick curves, where $|e(k)| < |e(k-1)|$. The reason is that when $e(k_1) = e(k_2)$ and the moving direction of $r_a(k_1)$ is away from CIR and the moving direction of $r_a(k_2)$ is towards CIR, a large $K_p$ at time $k_2$ may lead to unnecessary increase or decrease in $CIR_{\text{Thresh}}$. We use $\beta$ to achieve this goal.

3.4.2.2 Using upper and lower bounds

We have mentioned that VS-ACT uses CIR as the lower bound of $CIR_{\text{Thresh}}$ and 2CIR as the upper bound. The goal of using the upper and lower bounds is similar to the goal of anti-windup [20] strategies in the classical control theory. It is possible that any further increase in $CIR_{\text{Thresh}}$ does not lead to any improvement in bandwidth assurance when $CIR_{\text{Thresh}}$ increases past a certain point. If the integration of $e(k)$ continues in this case, the value of $CIR_{\text{Thresh}}$ becomes very large without any performance improvement. $e(k)$ then has to be of the opposite sign for a long time to bring the value of $CIR_{\text{Thresh}}$ back to its steady state value when the network conditions are changing. Thus, if there is no upper bound, there may exist an adverse impact on other aggregates improving bandwidth assurance and there may be an adverse impact on the fair sharing in excess bandwidth among aggregates when the network conditions are varying. In addition, when $CIR_{\text{Thresh}}$ or mp is allowed to be zero, the performance of the aggregate itself in achieving bandwidth assurance and sharing in excess bandwidth is degraded in some situations. The simulations in Section 3.5 illustrate the importance of using upper and lower bounds. The multiplicative-decrease feature in the TCP congestion control algorithm motivates choosing 2CIR as the upper bound.

3.4.2.3 Setting control parameters

The settings of $k_{\text{max}}$, $k_{\text{pmin}}$ and $k_{\text{umin}}$ are critical to the performance of the system. Using the conditions derived in Section 3.3 to derive $k_{\text{max}}$, $k_{\text{pmin}}$ and $k_{\text{umin}}$, if not impossible, is hard work, especially when there are a large number of aggregates involved. However, these conditions provide some theoretic guidelines for selecting $k_{\text{max}}$, $k_{\text{pmin}}$ and $k_{\text{umin}}$. When we choose $k_{\text{pmin}}$ and $k_{\text{umin}}$, we ignore the transient behavior and focus on the steady state
behavior. In the sufficient conditions derived for stability, the upper bounds are approximately decreasing functions of RTT and increasing functions of $N$ (the number of micro-flows of an aggregate). Thus, letting $k_{\text{max}}=0$, we choose $k_{\text{pmin}}$ and $k_{\text{min}}$, in the scenario, where (i) each adaptive aggregate has the same characteristics; (ii) the minimum $N^-$ and the large propagation delay $\text{RTT}^+$ are used; (iii) CIR is set to the average of the possible values used in all the simulations; (iv) setting the sending rate of non-adaptive aggregates such that the subscription level of the bottleneck link is light. The subscription level of a link is defined as the ratio of the sum of CIRs of adaptive aggregates and the sending rates of non-adaptive aggregates to the link bandwidth. We repeat simulations to find out the large values of $k_{\text{pmin}}$ and $k_{\text{min}}$ such that the goodput of each unsatisfied aggregate can approximate CIR. The motivation here is that the system with this VS-ACT at ingress routers is locally stable in the range of $\bar{N} \geq N^-$ and $\text{RTT} \leq \text{RTT}^+ \text{ when the system (} \bar{N}^-,\text{RTT}^+) \text{ with VS-ACT, using } (k_{\text{pmin}}, k_{\text{min}}) \text{ and } k_{\text{max}}=0 \text{, is stable.}$

We choose the value of $k_{\text{max}}$ in order to speed up the transient response without sacrificing the stability. We set the ability of grabbing bandwidth of each aggregate different in a large degree (we do it by assigning them with different propagation delay or with different CIRs) and the subscription level of the bottleneck link is heavy, such as the network scenarios in 3.5.1.3 and 0. We repeat simulations until the transient response is satisfactory while the steady state behavior is satisfactory.

### 3.5 Simulation results

We use ns-2 [74] to evaluate the effectiveness of VS-ACT and compare its performance with TSW, PME, MBM, ACT, and ARM.

The network topology used for simulations is shown in Figure 3.12. In this figure $S/D_i \ (1 \leq i \leq 10)$ is source/destination node; $I_i$ is ingress router; $E_i$ is egress router; and $C_i$ is core router. The link delay between $E_1$ and $D_i$ is 10ms. The capacities and delays of other links are set to 20Mbps and 5ms, respectively. There are 10 aggregates ($A_1-A_{10}$), where $A_i$ is from $S_i$ to $D_i$. An adaptive aggregate is defined as consisting only of identical adaptive micro-flows, which respond to congestion. A non-adaptive aggregate is defined as consisting only of identical non-adaptive micro-flows, which do not respond to
congestion. We summarize the attributes of each aggregate in Table II. We employ UDP sources sending constant bit rate (CBR) traffic as an example of non-adaptive sources. The sending rates of A_0 and A_{10} are both 5.0Mbps. We use TCP sources generating infinite FTP bulk data as adaptive sources. The TCP sources are based on the TCP-Reno implementation. C_1—E_1 is the bottleneck link and it is implicitly over-subscribed. The subscription level is 120%. Here, an under-subscribed (exact-subscribed) link is referred to as the link where the sum of CIRs of all competing aggregates is less than (equal to) the link capacity; an implicit over-subscribed link refers to a kind of under-subscribed links where the sum of CIRs of adaptive aggregates and the sending rates of non-adaptive aggregates is larger than the link capacity.

Some notations and the corresponding parameters used in the following section are defined in Table III. In PME, we find that when η is set to 0.045 PME performs better than setting other values to η in the following simulations. We use the time sliding window (TSW) profiler at the ingress routers when doing simulations with TSW, PME, MBM, ACT and VS-ACT and we use the token bucket profiler when doing simulations with ARM. The input of PME, MBM, ACT, and VS-ACT is computed by using the Exponentially Weighted Moving Average (EWMA) technique with the 1-second period and the weight of 0.8. The arriving rate in the 1-second period is computed by
Table II  Attributes of aggregates

<table>
<thead>
<tr>
<th>Aggregate</th>
<th># of micro-flows</th>
<th>Packet size (bytes)</th>
<th>CIR (Mbps)</th>
<th>Round Trip Propagation Delay (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptive aggregate</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A1</td>
<td>5</td>
<td>1000</td>
<td>2.0</td>
<td>50</td>
</tr>
<tr>
<td>A2</td>
<td>5</td>
<td>1000</td>
<td>2.0</td>
<td>50</td>
</tr>
<tr>
<td>A3</td>
<td>5</td>
<td>1000</td>
<td>2.0</td>
<td>50</td>
</tr>
<tr>
<td>A4</td>
<td>5</td>
<td>1000</td>
<td>2.0</td>
<td>50</td>
</tr>
<tr>
<td>A5</td>
<td>5</td>
<td>1000</td>
<td>2.0</td>
<td>50</td>
</tr>
<tr>
<td>A6</td>
<td>5</td>
<td>1000</td>
<td>2.0</td>
<td>50</td>
</tr>
<tr>
<td>A7</td>
<td>5</td>
<td>1000</td>
<td>2.0</td>
<td>50</td>
</tr>
<tr>
<td>A8</td>
<td>5</td>
<td>1000</td>
<td>0.0</td>
<td>50</td>
</tr>
<tr>
<td>Non-Adaptive aggregate</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>A9</td>
<td>1</td>
<td>1000</td>
<td>2.0</td>
<td>50</td>
</tr>
<tr>
<td>A10</td>
<td>1</td>
<td>1000</td>
<td>0.0</td>
<td>50</td>
</tr>
</tbody>
</table>

Table III  Schemes

<table>
<thead>
<tr>
<th>Mechanism</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>TSW</td>
<td>Time Sliding Window Two Color Marker</td>
</tr>
<tr>
<td>MBM</td>
<td>Memory-based Marking</td>
</tr>
<tr>
<td>PME</td>
<td>Packet Marking Engine, $\eta = 0.045$</td>
</tr>
<tr>
<td>ACT</td>
<td>Adaptive CIR Threshold, $\gamma = 0.05$, $\beta = 0.025$</td>
</tr>
<tr>
<td>VS-ACT</td>
<td>$k_{\text{min}} = 0.03$, $k_{\text{min}} = 0.028$, $k_{\text{max}} = 0.03$, $\eta = 0.5$Mbps, $\beta = 75%$</td>
</tr>
</tbody>
</table>

measuring the number of arriving packets over 1-second. We use 1.0 second as the sampling interval to update the marking probability in PME and 2.0s as the time interval to adjust $CIR_{\text{thresh}}$ in ACT and VS-ACT. The reason for choosing 2.0s is that the maximum RTT is in (1.0, 2.0)s in experiment 3.5.1.4. The reason for using 1.0s rather than 2.0s in PME is that the transient response in most experiments is too slow if we use 2.0s. For ARM, the parameters used in the controller are set as suggested in [14]. In [14], the sampling interval for adapting the token bucket rate is set to $1/37.5$s and the time interval for computing the average arriving rate is set to 1.0s. RIO is used as the AQM for all the following simulations. Note that the authors in [14] evaluate ARM by applying the two-level PI controller as AQM, which is different from my thesis. The RIO parameters, $[q_{\text{min}}, q_{\text{max}}, p_{\text{max}}]$ for IN and OUT, are set to $[150, 400, 0.02]$, $[80, 150, 0.1]$, respectively.
Adaptive hosts and network routers are ECN-enabled. The packet size at routers is 1000 bytes. Unless otherwise specified, the above settings are used as default values in the following simulations.

In the following we consider both static and dynamic scenarios. By static networks, we mean that network configurations and traffic characteristics is not changed during the whole simulation. The simulations in the static scenarios aim to examine the steady state behavior of the various mechanisms. The performance metric is the Average Goodput, computed by measuring the number of packets received at the receiver over a specified time period after the network is in the quasi-stable state. The simulations in the dynamic scenarios aim to examine the transient behavior of the various mechanisms. The performance metric is the Average Goodput (estimated per 5.0 seconds) variation in the simulation period.

3.5.1 Static network scenarios: under-subscribed

The simulations in this section examine the steady state performance of each scheme in under-subscribed networks. So far we have assumed that all the micro-flows in an aggregate are identical. Thus, when we exclude the impact of non-adaptive aggregates, the main elements that affect the ability of an adaptive aggregate in achieving bandwidth assurance are (i) the number of micro-flows in the aggregate; (ii) CIR of the aggregate; (iii) micro-flow characteristics such as packet size and RTT. We study the impact of each of these attributes on the Average Goodput. We vary one attribute at a time and examine the performance. The range of RTTs, packet sizes and CIRs is chosen according to the simulations in [19]. All the aggregates, A1-A10, are active. Each simulation lasts 800s. The Average Goodput of each aggregate in one simulation is computed from the 400th second to the 800th second. Each simulation is repeated 10 times, and then a final average is taken over all the runs. In the following, we first present the results for the various cases and then give remarks.

3.5.1.1 Impact of the number of micro-flows

The number of micro-flows of A1-A8 is set to 5, 10, 15, 20, 25, 30, 35 and 15 respectively. Other settings are same as in Table II. Figure 3.13 shows the Average
Figure 3.13  Static under-subscribed network --- Impact of # of micro-flows

Goodput achieved by A1-A7 for each scheme. In the figure, the horizontal line (at 2 Mbps) denotes the target rate to be achieved by each aggregate and 1-7 denote A1-A7, respectively.

3.5.1.2 Impact of packet size

The packet sizes of A1-A7 are set to 100 bytes, 300 bytes, 500 bytes, 700 bytes, 1000 bytes, 1200 bytes, 1500 bytes, respectively. Other settings are same as in TABLE II. Figure 3.14 shows the Average Goodput achieved by A1-A7 for each scheme. In the figure, the horizontal line (at 2 Mbps) denotes the target rate to be achieved by each aggregate and 1-7 denote A1-A7, respectively. B represents bytes.

3.5.1.3 Impact of target rate

The target rates of A1-A7 are set to 0.5 Mbps, 1 Mbps, 1.5 Mbps, 2 Mbps, 2.5 Mbps, 3.5 Mbps and 4.5 Mbps, respectively. So the subscription level is 127.5%. Other settings are same as in Table II. Figure 3.15 shows the Average Goodput Deviation of A1-A7 for each scheme. Average Goodput Deviation is defined as [(Average Goodput) – CIR]. Ideally the Average Goodput Deviation should be zero, as represented by the dashed line in the figure.
3.5.1.4 Impact of RTT

We set RTTs of A$_1$-A$_7$ to different values by setting the link delay of E$_i$—D$_i$ (i from 1 to 7) to 10ms, 50ms, 200ms, 350ms, 500ms, 650ms, and 800ms, respectively. Other settings are same as in Table II. Figure 3.16 shows the Average Goodput achieved by A$_1$...A$_7$ for each scheme. In the figure, the horizontal line (at 2 Mbps) denotes the target rate to be achieved by each aggregate and 1-7 denote A$_1$-A$_7$, respectively. We repeat the simulations by varying the link delay of S$_i$—I$_i$ (i from 1 to 7) instead of the link delay of E$_i$—D$_i$ to set the RTTs of different aggregates to different values. Similar results are obtained. We don’t show the results.
3.5.1.5 Remarks

The results in Figure 3.13—Figure 3.16 show that: (i) When MBM is employed, the Average Goodput of most AF adaptive aggregates in the four experiments can’t approximate CIR. Compared to TSW, the Average Goodput of some AF adaptive aggregates is improved, but some is degraded. Consistent with the analysis in Section 3.3.2.2, there is large difference in the Average Goodput among A1-A7. (ii) Compared to TSW and MBM, PME results in smaller difference in the Average Goodput among A1-A7, consistent with the analysis in Section 3.3.2.1. (iii) Due to the fixed control gains, ARM behaves better under some conditions but worse under other conditions. (iv) ACT and VS-ACT perform better than other mechanisms in the four experiments in the term of improving bandwidth assurance. When ACT is applied, A1-A7 can achieve their CIRs in Experiments 3.5.1.1-3.5.1.3. But in Experiment 0, the Average Goodput of A7 is far below its CIR while other conditionally-satisfied aggregates (A1-A6) obtain more than their own CIRs. We use \( \text{SumCIR}_{\text{Thresh}} \) to denote the sum of \( \text{CIR}_{\text{Thresh}} \) of all the aggregates passing through the bottleneck link. Figure 3.17 (a) and (b) give the \( \text{SumCIR}_{\text{Thresh}} \) variations in the four experiments of ACT scheme and VS-ACT scheme, respectively. Figure 3.17 (a) explains the performance of ACT in 0, validating the analysis in Section 3.3.3.1. Figure 3.17 (b) shows that, when VS-ACT is applied, the \( \text{SumCIR}_{\text{Thresh}} \) of VS-ACT is smaller than that of ACT. Thereby A7 in Experiment 0 has a greater chance to
increase its goodput. The Average Goodput of $A_1$-$A_7$ is very close to their CIRs in the four experiments.

### 3.5.2 Static network: exact-subscribed

Now we investigate the steady state performance of each scheme in the static exact-subscribed network. Only $A_1$-$A_7$ are active. The target rates of $A_1$-$A_7$ are set to 8.0 Mbps, 4.5 Mbps, 2.5 Mbps, 2.0 Mbps, 1.5 Mbps, 1.0 Mbps, and 0.5 Mbps, respectively. Other settings are the same as in Table II. The simulation lasts 800s. Figure 3.18 shows the Average Goodput Deviation of $A_1$-$A_7$ for each scheme. Ideally the Average Goodput Deviation should be zero, as represented by the dashed line in the figure. The Average Goodput at the receiver is computed from the 300th's to the 800th's. Each simulation is repeated 10 times, and then a final average is taken over all the runs. The results of this experiment further confirm the conclusions about TSW, MBM, ACT and VS-ACT made in 3.5.1. In this experiment, PME and ARM perform better than in Section 3.5.1. The reason is that the lower bounds of $mp$ and $CIR_{\text{Thresh}}$ are both 0.0.
3.5.3 Dynamic networks

The results in Experiments 3.5.1 and 3.5.2 display the failure of TSW, MBM and PME in providing bandwidth assurance in static networks. From this section onwards, we focus on evaluating ACT, ARM and VS-ACT by examining their transient behaviors. We have mentioned earlier that one factor degrading the performance of ACT is the slow decrease in $K_p$ of the *conditionally-satisfied* aggregates with smaller CIR or with larger $r_s$ when $r_s >$CIR. In Section 3.5.3.1, we examine the case of “with smaller CIR”, that is, the impact of the *conditionally-satisfied* aggregates with smaller CIR on the performance of the *unsatisfied* aggregate. In Section 3.5.3.2, we examine the case of “larger $r_s$”. We also use the simulations in 3.5.3.1 and 3.5.3.2 to show ARM performs better in some network situations but performs worst in other situations.

3.5.3.1 Varying non-adaptive traffic load

The network is dynamic due to the varying non-adaptive traffic load, which leads to the varying subscription level. The simulation lasts 800s. The sending rates of $A_9$ and $A_{10}$ are both 0.5 Mbps in $[0, 200^{th}]$s, 5.0Mbps in $[200^{th}, 400^{th}]$s, 1.0Mbps in $[400^{th}, 600^{th}]$s and 9.0Mbps in $[600^{th}, 800^{th}]$s, respectively. The target rates of $A_1$-$A_7$ are set to 7.0, 4.0, 3.0, 2.0, 1.5, 1.0 and 0.5 Mbps, respectively. $A_8$, $A_9$ and $A_{10}$ send best-effort traffic. Other settings are same as in Table II. Figure 3.19 (a), (b) and (c) depict the Average Goodput
variation of A₁-A₇ for ACT, VS-ACT and ARM, respectively. They are obtained by measuring the number of packets per 5 seconds. Figure 3.20 plots CIR_{thresh} variation and Average Goodput variation of A₁ for ACT and VS-ACT. CIR_{thresh_A₁_ACT} (CIR_{thresh_A₁_VS-ACT}) represents the variation of CIR_{thresh} of A₁ when ACT (VS-ACT) is applied. GB_{A₁_ACT} (GB_{A₁_VS-ACT}) represents Average Goodput variation of A₁ when ACT (VS-ACT) is applied.

The result in Figure 3.19 (a) shows that, when ACT is employed, the slowly-decreasing method damages the benefit of A₁ when the network is changing from a heavy implicit over-subscribed situation [200^{th}, 400^{th}]s to a light implicit over-subscribed situation [400, 600]s. In the whole simulation, A₂-A₇ can achieve their CIRs; but the goodput of A₁ can reach its CIR only in [0, 200^{th}]s and [500^{th}, 600^{th}]s. The reason is that in [200^{th}, 400^{th}]s excessive IN traffic in the network increases the probability of ECN-marking or dropping of the IN packets. Therefore, no matter how the CIR_{thresh} of A₁ is increased, when the sending rate of A₁ (all are IN packets) increases past a certain point, some IN packets of A₁ are ECN-marked or dropped. As a consequence, the Average Goodput of A₁ can’t reach its CIR. Figure 3.20 shows this. In [400^{th}, 600^{th}]s, the sending rates of A₉ and A₁₀ are small. Actually, A₁-A₇ can achieve their CIRs without using such a large CIR_{thresh} as in the previous periods. But Figure 3.19 (a) shows that, in [400^{th}, 500^{th}]s, the Average Goodput of A₁ is less than its CIR. This is because, during this period, A₂-A₇ slowly decrease the large value of their CIR_{thresh}, which is accumulated in [200^{th}, 400^{th}]s. Thus, there is still excessive IN traffic entering the network, preventing A₁ from increasing its goodput. A₁ in [600^{th}, 800^{th}]s behaves as in [200^{th}, 400^{th}]s.

Figure 3.19 (b) and (c) show that when VS-ACT and ARM are applied, the transient behavior is quite satisfactory in terms of the small settling time and the small overshoots. The Average Goodput of A₁-A₇ approximates their corresponding CIRs during the entire simulation.

3.5.3.2 Varying number of micro-flows in aggregates

In this experiment, we examine the case of “larger rₙ”. We vary the number of micro-flows in the aggregates in order to give them different abilities in grabbing bandwidths.
Figure 3.19  Average Goodput variation of A₁---A₇

Figure 3.20  Goodput and $CIR_{\text{Thresh}}$ variations of A₁
The number of micro-flows of A1-A8 is set to 5, 10, 15, 20, 25, 30, 35 and 15, respectively. All the aggregates are active. In order to avoid the impact of CIR, the CIRs of A1-A7 are all set to 2.5Mbps. The CIR of A9 is 1.5Mbps. The sending rates of A9 and A10 are both 5Mbps. Other settings are same as in Table II. The simulation lasts 600 seconds. In the first 200 seconds, only 5 micro-flows in each adaptive aggregate of A1-A7 are active. From the 200th to 400th, all micro-flows are active. In the last 200s, only 5 micro-flows in each adaptive aggregate are active. Figure 3.21 depicts the Average Goodput and CIR\text{Threshold} variations of A1 and A7 for ACT, VS-ACT and ARM.

We can see that: (i) when ACT is applied, during the first 200 seconds, A1 and A7 have the same characteristics and the CIR\text{Threshold} of both aggregates is approximately the same. At the 200th second, a number of micro-flows start. Although CIR\text{Threshold} of A1 is increased quickly (because $\gamma = 0.05$), due to the slow decrease in CIR\text{Threshold} of other aggressive conditionally-satisfied aggregates such as A7, the goodput of A1 is far below its CIR. This continues until the CIR\text{Threshold} of A7 is decreased sufficiently at about the 350th second. At the 400th second, most micro-flows stop and A1---A7 have the same traffic characteristics again. Thus, A1 can achieve its CIR without using so large CIR\text{Threshold} as in the previous period. But the slowly-decreasing method in ACT makes CIR\text{Threshold} of A1 decrease very slowly, delaying other aggregates such as A7 from improving bandwidth assurance. (ii) When VS-ACT is applied, it shows fast response to network changes and there is small variation in the Average Goodput of each adaptive AF aggregate in the whole simulations. (iii) Figure 3.21 (c) shows the weird performance of ARM during [200th, 400th]s. This is due to the fixed controller gains.

3.5.4 The importance of upper bound and lower bound on CIR\text{Threshold}

This experiment aims to investigate the importance of using upper bound and lower bound on CIR\text{Threshold}, mentioned in Section 3.4.2.2. We do simulations only with VS-ACT. A1-A7 have the same traffic characteristics as in Table II. The simulation lasts 800s. The sending rates of A9 and A10 are varying during the simulation, both 0.0 Mbps in [0, 200th]s, 9.0Mbps in [200th, 400th]s, 1.0Mbps in [400th, 600th]s and 9.0Mbps in [600th, 800th]s. We do two simulations and then compare their results. In the first
simulation, during the first 400s $CIR_{\text{Thresh}}$ varies in $[0.0, 2CIR]$; during the left 400s $CIR_{\text{Thresh}}$ varies in $[0.0, \infty]$. In the second simulation $CIR_{\text{Thresh}}$ is allowed to vary in $[CIR, 2CIR]$ during the entire simulation. We use the results in the first 400s to illustrate the importance of the lower bound and the results in the left 400s to investigate the importance of the upper bound. We only show the results of $A_1$, $A_8$, $A_9$, and $A_{10}$. Figure 3.22 (a) and (b) plot the Average Goodput variation of $A_1$ and $A_8$ in the two simulations, respectively. Figure 3.23 (a) and (b) plot the Average Goodput variation of $A_9$ and $A_{10}$ in the two simulations, respectively.

Figure 3.22 (a) shows that in $[80^{th}, 200^{th}]s$ and $[500^{th}, 600^{th}]s$, the Average Goodput of $A_1$ approximates the Average Goodput of $A_8$. Figure 3.22 (b) shows that when $CIR_{\text{Thresh}}$ is
Figure 3.22  Performance of $A_1$ and $A_8$

Figure 3.23  Performance of $A_9$ and $A_{10}$

bounded, the fairness in sharing excess bandwidth between $A_1$ and $A_8$ is improved greatly. In addition, the results in $[200^{th}, 400^{th}]$s and $[600^{th}, 800^{th}]$s show, when bounded, the bandwidth assurance of $A_1$ is achieved. Same conclusions can be made about non-adaptive aggregates $A_9$ and $A_{10}$ from the results in Figure 3.23.

3.6 Discussion and conclusion

In this chapter, we systematically explore the application of feedback control theory to design mechanisms in order to improve bandwidth assurance based only on the
knowledge gathered at ingress routers. We use a control theoretic approach to analyze some existing adaptive mechanisms in the literature. Then, a Variable-Structure PI controller for adapting CIR Threshold is developed. The performance evaluation results support the conclusions derived from our control-theoretic analysis of the existing algorithms and demonstrate the superiority of VS-ACT over a wide range of network dynamics.

As in the case of other ingress-based mechanisms that use only local knowledge to improve bandwidth assurance, VS-ACT also faces the problem of low domain throughput [13] when there exist aggressive non-adaptive flows. This problem can be alleviated by combined with the mechanism developed in [13].

Note that the ingress-based mechanisms discussed above run at the output queue at the ingress routers. All the above discussions in the under-subscribed networks assume that the arriving rate of an aggregate at the ingress input link card is equal to the departure rate of this aggregate from the ingress output link card. But this may not be true when switches, such as CIOQ switches, have multiple input and output queues. In such switches the existence of cross traffic between multiple input and output interfaces may cause the difference between the arriving rate and the departure rate of an aggregate and then affect the attainment of bandwidth assurance. When the failure of bandwidth assurance is caused by only cross-traffic, increasing $CIR_{\text{thresh}}$ contributes nothing to the attainment of bandwidth assurance. The authors in [6] propose a solution to prevent the failure of bandwidth assurance caused by cross-traffic. This solution and VS-ACT are complementary and can be used in conjunction with each other. Note that when the failure of bandwidth assurance of an aggregate is caused by cross-traffic inside the switch, increasing $CIR_{\text{thresh}}$ of this aggregate does not lead to serious performance degradation to those other flows that share other switches with this aggregate in the networks. By serious performance degradation, we mean that undesired increase of $CIR_{\text{thresh}}$ may cause transient performance degradation to other flows when there is no impact from cross-traffic, but the ingress-based mechanisms can quickly correct the undesired increase.
CHAPTER 4

A STABLE QUEUE-BASED ADAPTIVE CONTROLLER FOR AQM

4.1 Introduction

It is preferable to design an adaptive PID controller for AQM in order to reduce the tradeoff between fast transient response and stability. This chapter focuses on system-state-based adaptive mechanisms for AQMs. LRED and API are such kind of AQM schemes. However, neither of them studies the robustness of their AQM schemes theoretically. Although the control parameters in API are selected based on a pre-defined set of \((N^-, R^+, C)\), API can not guarantee system stability in the range of \((N^> N^-, R^< R^+)\), where a fix-gain PI can guarantee. We illustrate this by the following example. Considering a scenario for API: the control parameters are selected at the equilibrium point \((N^-, R^+, C)\) and assume \(N_1 >> N_2 > N^-\). When the network state is changing from \((N^-, R^+, C)\) to \((N_1, R, C)\), the stability is guaranteed. But from \((N_1, R, C)\) to \((N_2, R, C)\), there is no stability guarantee. That is, API can not assure system stability in the range of \((N^> N^-, R^< R^+)\). For LRED, the guarantee of system stability completely depends on the correct estimation of the packet loss ratio.

This chapter presents a stable queue-based adaptive controller and its implementation. We validate our design and compare its performance with some other AQM mechanisms through extensive simulations. The metrics for assessing the performance include: (i) queue fluctuations; (ii) bottleneck link utilization; (iii) robustness with respect to the unmodelled disturbance such as short-lived flows (popularly known as Web mice).

The rest of this chapter is organized as follows. In Section 4.2 we review the linearized control system previously developed in [33] and derives the conditions for a queue-based adaptive mechanism to stabilize the system. In Section 4.3, we present Q-SAPI. Then the experimental evaluation of Q-SAPI and comparison with other mechanisms are presented in Section 4.4. Finally we present the summaries in Section 4.5.
4.2 Stability condition for queue-based adaptive PI controller

The objective of designing queue-based adaptive AQM is to achieve satisfactory transient response without sacrificing the small steady state error in the range where \( N \geq N^- \) and \( R_0 \leq R^+ \). Here, \( N^- \) and \( R^+ \) are fixed values; \( N \) and \( R_0 \) are variables. In this section, we first state the general properties of the fluid-flow model of TCP congestion control algorithms together with AQM controller. Then, we derive the system stability conditions for using PI controller with varying gains for AQM.

4.2.1 The existing linearized AIMD/AQM system model

We assume that (i) there is only a single bottleneck link with capacity \( C \), (ii) there are \( N \) homogeneous TCP connections, (iii) TCP timeout and slow-start mechanism are not taken into account, and (iv) \((W_0, p_0, R_0)\) is the equilibrium point in the congestion avoidance phase. Then, Eq.(4.1) describes the linear relations among congestion window size \( W(t) \) (packets), round trip delay \( R_0 \) (sec), number of TCP connections \( N \), link capacity \( C \) (packets/sec), packet dropping probability \( p(t) \) and queue length \( q(t) \) around the equilibrium point \((W_0 = \frac{R_0 C}{N}, p_0 = \frac{2N^2}{R_0 C^2}, R_0)\) [61][33]. \( R_0 \) is the equilibrium value for round-trip delay \( R \). \((\delta W, \delta q)\) is perturbation from the equilibrium point.

\[
\begin{align*}
\delta \dot{W} &= \frac{2N}{C^2 R_0^2} \delta W - \frac{2N}{2N} \delta p(t - R_0) \\
\delta \dot{q} &= \frac{N}{R_0} \delta W
\end{align*}
\] (4.1)

The block diagram of the AIMD/AQM system is shown in Figure 4.1, where \( \tilde{G}(s) = \frac{K e^{-\tau}}{a_1 s^2 + a_2 s + 1} \) is the linear "plant" and \( C(s) \) is the controller for AQM. In \( \tilde{G}(s) \), \( K = \frac{C^3 R_0^3}{(2N)^3} \) and \( a_1 = \frac{R_0^2 C}{2N} \) and \( a_2 = R_0 + \frac{R_0^2 C}{2N} \). The details of linearizing can be found in [33].

Here AIMD denotes additive increase and multiplicative decrease.
Eq.(4.2) depicts the continuous-time Proportional-Integral control (PI) controller, where \( e(t) = q(t) - q_{ref} \) and \( p(t) \) is the packet dropping probability at time \( t \).

\[
p(t) = k_p \times e(t) + k_i \times \int e(t) \tag{4.2}
\]

4.2.2 Motivation for varying gains according to queue length error

In the model given by Eq.(4.1), changes in \( N \) or \( C \) or \( R \) suggest that network state is changing. The changing network state often leads to the change in the steady state dropping probability. Before the network reaches the new equilibrium point, there are often queue fluctuations. When we assume congestion always exists and ignore disturbance, \( |e(t)| \) can be an indication of the magnitude of this change. The proportional control action changes \( e(t) \) in proportion to \( |e(t)| \) and in the direction, which reduces \( |e(t)| \). The integral action changes \( e(t) \) incrementally, in proportion to the time integral of previous errors [3]. These serve as the motivation for designing \( k_p \) and \( k_i \) as increasing functions of \( |e(t)| \) so as to produce faster transient response and alleviate the possible instability, caused by using large constant \( k_p \) and \( k_i \) to speed up transient response.

4.2.3 Stability conditions for PI controller with varying gains

In this subsection, we derive the conditions for a PI controller with varying gains, adjusted according to \( |e(t)| \), to stabilize the system described in Eq.(4.1) in the range of \( N \geq N^- \) and \( R \leq R^+ \). We use \( G(s) \) to denote the transfer function of the plant when the network parameters are set to \( (N^-, R^+, C) \). Thus, \( G(s) = \frac{(CR^+)^3 e^{-MR^+}}{(2N^-)^3 s^3 + (R^+)^3 C s^2 + \left( R^+ + \frac{(R^+)^2 C}{2N^-} \right) s + 1} \).

We use \( \tilde{G}(s) \) to denote the transfer function of the plant when the network parameters are
in the range of \( N > N^- \), \( R_0 < R^+ \). Thus, \( \tilde{G}(s) = \frac{(CR_0)^3 e^{-R_0}}{(2N)^2 s^2 + \left( \frac{(R_0)^3 C}{2N} + (R_0)^3 C \right) s + 1} \). In the following, if we denote the plant of a system by \( G(s) \), we call the system as system \( G(s) \). So is system \( \tilde{G}(s) \).

**Theorem 1** If a fixed-gain PI controller \( C(s) = k_{p\min} + \frac{k_{i\min}}{s} \) for AQM can stabilize system \( G(s) \), then this controller can stabilize system \( \tilde{G}(s) \).

The proof is given in [31].

**Theorem 2** There must exist a positive \( k_{i\text{incr}} \) such that when \( k_p(t) \) is a continuous, memoryless and time-invariant function of \( |e(t)| \) and is varying in \( [k_{p\min}, k_{p\min} + k_{i\text{incr}}] \), the controller \( C(s) = k_p + \frac{k_{i\min}}{s} \) asymptotically stabilizes system \( G(s) \).

**Proof:** We rearrange the components in Figure 2.1 and obtain its equivalent in Figure 4.2 such that the linear part \( T_1(s) \) is kept in the forward path and the nonlinear part \( k_p - k_{p\min} \) is placed in the feedback loop. In Figure 4.2, \( T_1(s) = G(s)/(1 + G(s)(k_{p\min} + k_{i\min}/s)) \). The assumption in Theorem 1 assures that \( T_1(s) \) is open-loop stable.

Define frequency \( \omega \in (-\infty, +\infty) \). Letting \( e^{j\omega R^+} = \cos(-\omega R^+) + j\sin(-\omega R^+) \) and assuming

\[
\begin{align*}
A &= -a_2 \omega^3 + Kk_{p\min} \cos(\omega R^+) + Kk_{p\min} \omega \sin(\omega R^+) \\
B &= \omega + Kk_{p\min} \omega \cos(\omega R^+) - Kk_{p\min} \sin(\omega R^+) - a_3 \omega^3
\end{align*}
\]

we obtain

\[
T_1(j\omega) = K \omega \frac{A \sin(\omega R^*) + B \cos(\omega R^*)}{A^2 + B^2}
\]

Letting \( \beta = \arctg(B/A) \) and letting \( \text{Re} \) and \( \text{Im} \) denote the real and imaginary parts of complex \( T_1(j\omega) \), respectively. Then we obtain
\[
\begin{align*}
\text{Re}\{T_1(j\omega)\} &= \frac{K\omega \sin(\omega R^* + \beta)}{\sqrt{A^2 + B^2}} \\
\omega \text{Im}\{T_1(j\omega)\} &= K\omega \frac{\cos(\omega R^* - \beta)}{\sqrt{A^2 + B^2}} 
\end{align*}
\]

(4.3)

It is clear that \(\text{Re}\{T_1(j\omega)\}\) and \(\text{Im}\{T_1(j\omega)\}\) are bounded for all \(\omega\). Thus, there must exist a pair of positive \((\hat{\mathcal{Q}}_1, \hat{k}_{\text{min}}^p)\) such that

\[
\left(\hat{k}_{\text{min}}^p \right)^{-1} + \text{Re}\{(1 + j\omega \hat{\mathcal{Q}}_1)T_1(j\omega)\} > 0 
\]

(4.4)

According to Popov stability criterion [64], the system consisting of \(G(s)\) and \(C(s) = k_p + \frac{k_{\text{min}}}{s}\) is asymptotically stable. The proof is done. □

**Theorem 3** The controller \(C(s) = k_p + \frac{k_{\text{min}}}{s}\), designed according to Theorem 2, stabilizes system \(\hat{G}(s)\).

**Proof:** Define \(\hat{T}_1(s) = \hat{G}(s)/(1 + \hat{G}(s)^*(k_{\text{pmin}} + k_{\text{min}}/s))\). On the \((\text{Re}\{\cdot\}, \omega \text{Im}\{\cdot\})\)-plane, we draw \((\text{Re}\{\hat{T}_1(j\omega)\}, \omega \text{Im}\{\hat{T}_1(j\omega)\})\) and \((\text{Re}\{T_1(j\omega)\}, \omega \text{Im}\{T_1(j\omega)\})\) when \(\omega\) is varying in \((-\infty, +\infty)\). Because the real and imaginary parts of both \(\hat{T}_1(j\omega)\) and \(T_1(j\omega)\) are all even functions, it is enough if we consider \(\omega\) only in \([0, +\infty]\). Denote by \(L_1\) the plot of \((\text{Re}\{T_1(j\omega)\}, \omega \text{Im}\{T_1(j\omega)\})\) from its first interaction with the negative vertical axis to its first interaction with the positive vertical axis when \(\omega\) is varying from zero to infinite; denote by \(L_1\) the plot of \((\text{Re}\{T_1(j\omega)\}, \omega \text{Im}\{T_1(j\omega)\})\) from its first interaction with the negative vertical axis to its first interaction with the positive vertical axis. It is obvious
that (i) the plot of \( \{\text{Re}\{\hat{T}_i(j\omega)\}, \text{Re}\{\text{Im}\{\hat{T}_i(j\omega)\}\}\} \) when \( \omega \) is from 0 to infinite is on the right side of \( \hat{L}_i \); same conclusion for \( L_1 \) and \( \{\text{Re}\{T_i(j\omega)\}, \text{Re}\{\text{Im}\{T_i(j\omega)\}\}\} \); (ii) \( L_i \) bounds \( L_i \) for all \( N > N^* \) and \( R_0 < R' \). Thus, \( \left( \hat{q} \right)_i, \hat{k}_{i_{incr}}^p \) must satisfy condition

\[
\left( \hat{k}_{i_{incr}}^p \right)^{-1} + \text{Re}\left( 1 + j\omega \hat{q} \right) \hat{T}_i(j\omega) > 0
\]

According to Popov stability criterion, the system consisting of \( \hat{G}(s) \) and \( C(s) \) is asymptotically stable. The proof is done.

**Theorem 4** There must exist a positive \( \hat{k}_{i_{incr}}^l \) such that when \( k_i(t) \) is a continuous, memoryless and time-invariant function of \( |e(t)| \) and varying in \( [k_{i_{imin}}^l, k_{i_{imin}}^l + \hat{k}_{i_{incr}}^l] \), the controller \( C(s) = k_{p_{min}} + \frac{k_i}{s} \) stabilizes system \( \hat{G}(s) \).

The proof of Theorem 4 is similar to Theorem 3.

**Theorem 5** Assume there exists positive real \( \left( k_{i_{incr}}^p, k_{i_{incr}}^l, q_1, q_2, \tau \right) \) satisfying conditions (4.5) (4.6), and (4.7). When \( k_p(t) \) and \( k_i(t) \) are both continuous, memoryless and time-invariant functions of \( |e(t)| \) and varying in \( [k_{p_{min}}, k_{p_{min}} + k_{p_{incr}}^p] \) and \( [k_{i_{imin}}, k_{i_{imin}} + k_{i_{incr}}^l] \), respectively, then the controller \( C(s) = k_p + \frac{k_i}{s} \) stabilize system \( G(s) \).

\[
\left( k_{i_{incr}}^p \right)^{-1} + \text{Re}\left( 1 + j\omega \hat{q} \right) \hat{T}_i(j\omega) > 0 \quad (4.5)
\]

\[
\left( k_{i_{incr}}^l \right)^{-1} + \text{Re}\left( 1 + j\omega \hat{q} \right) \hat{T}_i(j\omega) > 0 \quad (4.6)
\]

\[
f(\tau, N, R) = A(\omega)\tau^l + B(\omega)\tau + C(\omega) < 0 \quad (4.7)
\]

where
\[
\begin{align*}
A(\omega) &= \text{Re}\left[(1+j\omega T_1(\omega))\right]^2 + \text{Im}\left[(1+j\omega T_1(\omega))\right]^2 \\
B(\omega) &= 2\left[\text{Re}\left[(1+j\omega T_1(\omega))\right]\text{Re}\left[(1+j\omega T_1(\omega))\right]
- \text{Im}\left[(1+j\omega T_1(\omega))\right]\text{Im}\left[(1+j\omega T_1(\omega))\right]\right] \\
- 4\left[\mu^2 + \text{Re}\left[(1+j\omega T_1(\omega))\right]\right]\left[\mu^2 + \text{Re}\left[(1+j\omega T_1(\omega))\right]\right] \\
C(\omega) &= \text{Re}\left[(1+j\omega T_1(\omega))\right]^2 + \text{Im}\left[(1+j\omega T_1(\omega))\right]^2 \\
T_1(\omega) &= K\left[A\cos(\omega R^*) - B\sin(\omega R^*)\right] - \frac{1}{A' + B'}
\end{align*}
\]

**Proof:** Define \(\phi_1 = (k_p(t) - k_{p_{\text{min}}})e(t)\) and \(\phi_2 = (k_i(t) - k_{i_{\text{min}}})e(t)\). Thus, both \(\phi_1/e(t)\) and \(\phi_2/e(t)\) are continuous, memoryless and time-invariant and sector-bounded [43].

Extend \(\phi\) and \(e\) to complex variables. We rearrange the components in Figure 4.1 and obtain its equivalent in Figure 4.3, which is further rearranged such that the linear part \(T(s)\) is kept in the forward path and the nonlinear part of \(\phi = (\phi_1, \phi_2)^T\) is placed in the feedback loop, shown in Figure 4.4. Here, \(T(s) = \begin{pmatrix} T_{11}(s) & T_{12}(s) \\ T_{21}(s) & T_{22}(s) \end{pmatrix}\). \(T(s)\) is from the input \(\phi\) to the output \(e\). Thus, \(e = -T^*\phi\). \(\phi\) and \(e\) are complex vectors.

Define \(\mu_1 = k_{i_{\text{min}}}\) and \(\mu_2 = k_{i_{\text{max}}}\). Define \(V_1 = \phi_1(\mu_1 e_1 - \phi_1)\) and \(V_2 = \phi_2(\mu_2 e_2 - \phi_2)\), where

\[
\mu_1 = \begin{pmatrix} \mu_1^1 & 0 \\ 0 & \mu_1^2 \end{pmatrix}, 0 < \frac{\phi_1}{e_1} < \mu_1, i = 1, 2
\]

Thus, \(V_1 > 0\) and \(V_2 > 0\). Defining \(\tau = \begin{pmatrix} \tau_1 & 0 \\ 0 & \tau_2 \end{pmatrix}\) where \(\tau_1\) and \(\tau_2\) are positive real variable.

A quadratic form \(V\) is defined as

\[
V(\phi, e) = \text{Re}\left[\tau_1\phi_1(e_1 - \mu_1^2\phi_1) + \tau_2\phi_2(e_1 - \mu_1^2\phi_1)\right]
= \text{Re}\left[\phi^T(-T(s) - \mu_1^2)\phi\right]
\]

(4.8)

Defining \([\bullet]^*\) is complex conjugate of \([\bullet]\) and defining a quadratic form \(F\) with coefficient depending on a complex parameter \(s\), shown in Eq.(4.9)

\[
F(s, \phi) = \text{Re}[s\phi^T \Theta e] + V(\phi, e)
\]

(4.9)

where

\[
\Theta = \begin{pmatrix} \theta_1 & 0 \\ 0 & \theta_2 \end{pmatrix}, \theta_j \text{ is positive real variable} > 0 (j=1,2)
\]

70
Define $\pi(j\omega) = \tau\mu^{-1} + (\tau + j\omega\theta)T(j\omega)$. We obtain $F(j\omega, \varphi) = -\varphi^*(\pi(j\omega) + \pi^*(j\omega))\varphi / 2$ and

$$
\pi(j\omega) = \begin{pmatrix}
(j\omega\theta_1 + \tau_1)T_1 + \tau_1\mu_1^{-1} & (j\omega\theta_1 + \tau_1)T_2 \\
(j\omega\theta_2 + \tau_1)T_1 & (j\omega\theta_2 + \tau_1)T_2 + \tau_2\mu_2^{-1}
\end{pmatrix}
$$

Define $\pi(j\omega) = \left(\begin{array}{cc}
\Gamma_1(j\omega) & \Gamma_2(j\omega) \\
\Gamma_3(j\omega) & \Gamma_4(j\omega)
\end{array}\right)$, we obtain

$$
\frac{\pi(j\omega) + \pi^*(j\omega)}{2} = \begin{pmatrix}
\text{Re}(\Gamma_1(j\omega)) & \text{Re}(\Gamma_2(j\omega)) + \text{Re}(\Gamma_3(j\omega)) - \text{Im}(\Gamma_5(j\omega)) - \text{Im}(\Gamma_7(j\omega)) \\
\text{Re}(\Gamma_3(j\omega)) + \text{Re}(\Gamma_5(j\omega)) + \text{Re}(\Gamma_7(j\omega)) - \text{Im}(\Gamma_5(j\omega)) - \text{Im}(\Gamma_7(j\omega)) & \text{Re}(\Gamma_4(j\omega))
\end{pmatrix}
$$

Thus, $\pi(j\omega) + \pi^*(j\omega)$ is a Hermite matrix. Define $\tau_1 / \tau_2 = \tau$ and $q_i = \theta_i / \tau_i$. According to Sylvester’s criterion [48], if there exists $(k_{inc}^p, k_{inc}^s, q_1, q_2, \tau)$ satisfying conditions (4.5), (4.6) and (4.7), $\pi(j\omega) + \pi^*(j\omega)$ is positive definite for all $\omega (-\infty \leq \omega \leq +\infty)$. That is, $F(j\omega, \varphi)$ is
negative definite for all $\omega (-\infty \leq \omega \leq \infty)$. According to Yakubovich Approach [81], the control system is asymptotically stable.

**Theorem 6** The controller $C(s) = k_p + \frac{k_i}{s}$, designed according to Theorem 5, must stabilize system $\tilde{G}(s)$.

**Proof:** $A(\omega)$ and $C(\omega)$ are both inversely related to $N$ and $1/R_0$. $B(\omega)$ is inversely related to $R_0$ and $1/N$. Thus, $f(\tau, N, R_0) < f(\tau, N^{-}, R^+) < 0$. Based on theorem 1-6, the proof is done. \qed

### 4.3 Implementation of a queue-based adaptive PI controller

In the following, we first present an implementation of the Q-SAPI controller, and then give some explanations.

#### 4.3.1 Description of the implementation

Figure 4.5 gives the $k_p(t)$ and $k_i(t)$ algorithms. $k_p(t)$ and $k_i(t)$ are both non-decreasing nonlinear functions of $|e(t)|$ and are bounded in the sectors of $[k_{\text{pmin}}, k_{\text{pmin}} + k_{\text{incr}}]$ and $[k_{\text{imin}}, k_{\text{imin}} + k_{\text{incr}}]$, respectively. Figure 4.6 shows the algorithm of computing the dropping/marketing probability in Q-SAPI at the time instant $t$.

#### 4.3.2 Selecting parameters: $k_{\text{pmin}}, k_{\text{imin}}, k_{\text{incr}}, k_{\text{incr}}$

Given the algorithms in Figure 4.5, the key problem is how to set the control parameters such that the closed-loop system is stable in the range of all $N \geq N^{-}$ and $R_0 \leq R^+$. The sufficient stability conditions in Theorem 6 provide guidelines. Consider $(N^-, R^+, C) = (60, 0.51s, 3750 \text{packets})$. Using the method in [31], we choose phase margin to 75°. Thus, $(k_{\text{pmin}}, k_{\text{imin}}) = (4.30096 * 10^{-6}, 5.2914 * 10^{-7})$. In order to find $(k_{\text{incr}}^p, k_{\text{incr}}^i, q_1, q_2, \tau)$ to satisfy conditions (4.5), (4.6) and (4.7), we first find out the range of $(k_{\text{incr}}^p, k_{\text{incr}}^i, q_1, q_2)$ satisfying conditions (4.5) and (4.6). This can be done by using a graphical analysis method [64], which is based on the geometric interpretation of Popov Criterion. Figure 4.7 and Figure 4.8 depict the modified Nyquist trajectory of $T_1(s)$ and $T_2(s)$. 

72
\( e(t) = q(t)-q_{ref} \)

If \( e(t) = -q_{ref} \), then \( e(t) = -q_{ref} - 1 \)

\[
x(t) = \begin{cases} 0 & \text{if } |e(t)| < k_0 \\ e(t) - k_0 & \text{otherwise} \end{cases}
\]

\[
k_p(t) = k_{p_{min}} + k_{p_{incr}} \left( \frac{2}{1 + \exp\left(-\frac{x(t)^2}{k_1}\right)} - 1 \right)
\]

\[
k_i(t) = k_{i_{min}} + k_{i_{incr}} \left( \frac{2}{1 + \exp\left(-\frac{x(t)^2}{k_2}\right)} - 1 \right)
\]

1) \( k_{p_{min}}, k_{p_{incr}}, k_{i_{min}}, k_{i_{incr}} \) are positive constants, selected to stabilize the closed-loop system.
2) \( k_0, k_1, k_2 \) are user-defined positive constants.

Figure 4.5 \( k_p(t) \) and \( k_i(t) \) algorithms

At each time instant \( n \)

1) Using the algorithm in Figure 4.5 to compute \( k_p(n), k_i(n) \)
2) \( p(n) = p(n-1) + k_p \times |e(n)-e(n-1)| + k_i \times e(n) \)
3) \( p(n) = \min\{\max\{p(n), 0\}, 1\} \)

Figure 4.6 Dropping/marking probability algorithm

Figure 4.7 Modified Nyquist trajectory and Popov straight line for \( T_1(s) \)
respectively, and their related Popov’s line. The solid lines are the related Popov’s lines; the curves are the modified Nyquist trajectory. We obtain $k_{incr}^p < 1/32000$, $q_1=3.0$, $k_{incr}^l < 1/200000$, $q_2=1.2$. Note that these ranges are conservative.

Now we choose $(k_{incr}^p, k_{incr}^l, q_1, q_2)$ from the corresponding ranges such that there exists a positive $\tau$ satisfying condition (4.7). In order to produce a fast transient response without sacrificing stability, we would like to pick up $(k_{incr}^p, k_{incr}^l)$ as large as possible. Note that there is not a best way to find out $(k_{incr}^p, k_{incr}^l, q_1, q_2)$. The following method is intuitive. It is based on the relations between $f$ and $(k_{incr}^p, k_{incr}^l, q_1, q_2)$. Because $A(\omega)$, $B(\omega)$, $C(\omega)$ are bounded in the range of $0 \leq \omega \leq \infty$, there must exist a curve enveloping all $f(\tau)$ curves in the $\tau$-$f$ plane from the upside. The envelope curve can be used to check whether condition (4.7) is satisfied. If there exists a negative point on the envelope, we say condition (4.7) is satisfied; otherwise, we could decrease the values of $k_{incr}^p$ and $k_{incr}^l$ to check the new value of $f(\tau)$. The motivation here is that $f(\tau)$ is an increasing function of $k_{incr}^p$ and $k_{incr}^l$, then there must exist the values of $k_{incr}^p$ and $k_{incr}^l$ to satisfy condition (4.7). In addition, the function $p(n)$ in (4.5) indicates that, when $e(n)$ is larger, $k_i$ contributes more than $k_p$. Thus, we try to choose $k_{incr}^l$ as large as possible. Because $A(\omega)$, $B(\omega)$ are both
increasing functions of $q_1$ and $q_2$, certain decrease in $q_1$ and $q_2$ without unsatisfying conditions (4.5) and (4.6) is helpful to make condition (4.7) satisfied with larger $k_{incr}^p$ and $k_{incr}^l$. Such way is also intuitive. Thus the selected $k_{incr}^p$ and $k_{incr}^l$ may be not the optimal. By optimal, we mean that the largest $k_{incr}^p$ and $k_{incr}^l$ are chosen and the system stability is maintained.

Figure 4.9 depicts the variation of $f(\tau)$ versus $\tau$ when $k_{incr}^p=1/89588$, $k_{incr}^l=1/29200$, $q_1=q_2=0.9999$. It is obvious that $f(0.208)<0$. It can be proved that $f(0.208)<0$ is true for all $\omega$.

**Remarks:** The above approach of selecting the upper bound of $k_{incr}^p$ and $k_{incr}^l$ in order to guarantee the asymptotic stability of the system is conservative [71]. That is, in some network scenarios we guarantee the system stability by sacrificing the transient performance. However, as we will see in the following simulations, Q-SAPI with such bounds could exhibit satisfactory transient response while producing small steady state error.

### 4.3.3 Discussions of $k_0$, $k_1$, $k_2$

The above discussions are based on the assumption that there is no disturbance. But burst, a kind of disturbance, is the nature of current network traffic. In addition, those flows, which finish transmission in slow-start phase, also produce a kind of disturbance to the system described by Eq.(4.1). In order to reduce the effect of these disturbances on the system performance, $k_p(t)$ and $k(t)$ are increased at a very slow speed when $|e(t)|$ is small. In Q-SAPI, Figure 4.6 shows that: (i) when $|e(t)|$ is varying in $[0, k_0]$, $k_p(t)=k_{p_{min}}$; (ii) when $|e(t)|$ is varying in $[k_0, k_1]$, $k_p(t)$ has a slow increase; (iii) when $|e(t)|$ is varying in $[k_1, q_{lim}-q_{ref}]$, $k_p$ has a fast increase. Figure 4.10 shows the variation of $k_p(t)$ as a function of $|e(t)|$ when $k_{p_{min}}=0.2$, $k_{incr}^p=1.0$, $k_0=k_1=50$. We could see that $k_p(t)$ is continuous increasing function of $|e(t)|$, which is required in Theorem 6. Similar explanations can also be applied to $k_l(t)$. In Figure 4.5, when queue length is zero, we set $e(k)=-q_{ref}=-1$. Such modification is also applied to the original implementations of PI, REM and LRED. The main reason is that when $q_{ref}=0$ and $q_{len}=0$, the original

75
implementations can not provide required decrease in dropping probability. Such modification does not affect the system stability.
4.4 Performance evaluation

In this section, we carry out simulations with ns-2 [74] to evaluate Q-SAPI. We compare its performance with PI, REM, AVQ and LRED. We choose these AQMs because either they are implemented in ns-2 or their authors give the detailed implementations. In addition, these mechanisms cover rate-based AQMs, queue-based AQMs, the combination of queue-based and rate-based, and adaptive AQM mechanisms. We ignore the simulations of RED because there are lots of studies on RED behaviors.

Unless otherwise noted, a dumbbell network topology in Figure 4.11 is used with a bottleneck link capacity of 15Mbps and packet size of 500bytes. Round-trip link delay is uniformly distributed in [60,1000]ms. Target queue length $q_{ref}$ is set to 100 packets in default. As in [31], the bottleneck buffer size is set to 800packets. Although round-trip link delay in the followings is in [60,1000]ms, we think it is inadequate to choose parameters for the worst case, such as selecting control parameters by setting RTT=1.0s. It would be more effective for an AQM when its control parameters are selected for average traffic conditions. The recent Internet measurement study in [39] indicates that the distribution of the median value of the RTT of 50% of the senders is less than 300-400ms. Thus, the control parameters in PI and AVQ and Q-SAPI are set according to the theorems in their papers with $N=60$ and RTT=0.51s. This may result in the system instability. Thus, the following experiment also illustrates the robustness and parameter sensitivity of each AQM scheme. The parameters of REM and LRED are set same as their papers. The main reason is that the authors in [4] did not provide the setting method and our simulation results show that LRED with $\beta=0.001$ performs better in the scenario with $N=60$ and RTT=0.51s and $C=3750$packets than with $\beta$ chosen based the theorems in [78]. The parameters of these AQMs are listed in Table IV.

We do simulations under (1) varying number of FTP flows, (2) varying round-trip link delay, (3) varying bottleneck link capacity and (4) multiple congestion bottleneck links. Each simulation has different numbers, specified in each experiment subsection, of forward and backward direction FTP flows. Forward FTP flows are from $S_i$ to $D_i$; backward FTP flows are from $D_i$ to $S_i$. In addition, each simulation has 300 background Web-like sessions (using the Web traffic code built into ns-2) that start evenly.
Figure 4.11 Network topology with single bottleneck link

<table>
<thead>
<tr>
<th>Controller</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>PI</td>
<td>(kp, ki)=(4.30096 *10^-6, 5.2914 *10^-7), T =1/160 s</td>
</tr>
<tr>
<td>AVQ</td>
<td>γ=0.98, α=0.06</td>
</tr>
<tr>
<td>REM</td>
<td>γ=0.0012, φ= 1.001, α=0.1, T =1/160 s</td>
</tr>
</tbody>
</table>
| Q-SAPI     | (k_{min}, k_{max})=(4.30096 *10^-6, 1.5463*10^-5)  
             | (k_{min}, k_{max})=(5.2914 *10^-7, 3.9538*10^-6), k0=k1=30, k 2=10, T =1/160 s |
| LRED       | β=0.001, T=1s |

distributed in the first 30 seconds. Based on the results in [34], each Web session requests pages with 3 objects drawn from a Pareto distribution with a shape parameter of 1.2 and an average size of 5KB. The Web sessions have an exponentially distributed idle time with a mean of 7 seconds, which results in an average utilization of 25% of the 15Mbps capacity, a typical fraction of Internet traffic as reported by [73]. In all the simulations, adaptive sources for both long-lived FTP flows and short-lived Web sessions are over TCP/Reno; the initial window size of each TCP connection is set to 500 packets. All adaptive connections and routers are ECN-enabled. Thus, the packet loss ratio in LRED is estimated based on the number of the ECN-marked and dropped packets. Performance is evaluated using the instantaneous queue length variation and bottleneck link utilization.
variation in the simulation period, which can reflect both transient and steady state behaviors of each scheme.

4.4.1 Varying FTP flows

This experiment compares the performance of each AQM scheme under the varying number of forward FTP flows. Each simulation begins with (i) 10 forward direction FTP flows, starting uniformly distributed in the first one second, (ii) 300 background Web sessions and (iii) 50 backward FTP flows, starting uniformly distributed in the first 50s. At the 100th, 200th, 300th, 400th, additional 40 FTP flows, 50 flows, 100 flows, 100 flows are added, respectively. From the 500th, 100flows, 100flows, 50 flows are stopped at the 500th, 600th, and 700th, respectively. Figure 4.12 depicts the queue variation versus time (left column) and bottleneck link utilization variation versus time (right column) under each AQM scheme.

In terms of high link utilization and stable queue delay, LRED and Q-SAPI produce similar performance under dynamic traffic load, better than other AQMs. PI has much slower response and can not reach the stable queue length within 100s when the number of forward FTP flows is increased to 200 and to 300. This slow transient response leads to large queue delay in [300th, 500th]s and low link utilization in [500th, 720th]s when some forward flows stop. REM also exhibits longer period of low utilization after the traffic load is changing at the 500th and at the 600th. This is mainly due to the excessive congestion notification. Figure 4.12 shows that the queue length under REM is below $q_{ref}$ although REM aims to stabilize queue length around $q_{ref}$. AVQ is rate-based AQM and it can not effectively control queue length around a fixed value in the dynamic networks.

It is common that the value of $q_{ref}$ affects the performance of those AQMs, which use queue length to detect congestion, such as PI, REM, LRED and Q-SAPI. In order to investigate the robustness, we repeat the above simulations by setting $q_{ref}=0$. Since AVQ is rate-based AQM, we ignore its simulations. Figure 4.13 depicts the results. We could see that Q-SAPI performs best in terms of small queue delay and high link utilization. LRED perform worst. It is due to its method of computing dropping probability. The higher link utilization of PI over Q-SAPI is at the cost of large queue delay.
The number of forward FTP flows

Queue length (packets) variation versus time (second)

Link utilization variation versus time (second)

Figure 4.12 Dynamic forward FTP flows with $q_{rel} = 100$ packets
4.4.2 Varying round-trip time

This experiment aims to investigate the ability of each scheme under different RTTs, varying from about 0.3s to 1.5s. All connections have the same RTT. Other setting and configurations are set as default. We achieve varying RTT by periodically increasing bottleneck link delay. When the delay increases, we don’t change parameter settings of each AQM. This may result in the system instability. Thus, the following experiment also illustrates the robustness of each AQM scheme.

Figure 4.14 depicts the queue variation versus time (left column) and bottleneck link utilization versus time (right column) for the five AQMs. AVQ and LRED have poor
control over queue length. LRED has both large queue fluctuations and low link utilization. AVQ produces high link utilization at the cost of large queue fluctuations. In terms of small queue fluctuations and high link utilization, Q-SAPI performs better than
REM. When RTT is beyond the range to guarantee the stable system, the varying controller gains in Q-SAPI makes the system more unstable.

4.4.3 Varying bottleneck link capacity

The experiments in this subsection aim to investigate the ability of each scheme under varying bottleneck link capacities. There are 100 forward FTP flows, 50 backward FTP flows and 300 background Web-like sessions. We first do simulations by varying capacity in $[10\text{Mbps}, 50\text{Mbps}]$. That is, the capacity is 10Mbps, 20Mbps, 30Mbps, 40Mbps, 50Mbps in $[0^{th}, 200^{th}]$s, $[200^{th}, 400^{th}]$s, $[400^{th}, 600^{th}]$s, $[600^{th}, 800^{th}]$s, $[800^{th}, 1000^{th}]$s, respectively. Figure 4.15 depicts the queue variation versus time (left column) and bottleneck link utilization versus time (right column) for the five AQMs.

Then, we set the number of forward FTP flows to 200 and do simulations by setting capacity to 10Mbps, 30Mbps, 50Mbps, 70Mbps, 100Mbps in $[0^{th}, 200^{th}]$s, $[200^{th}, 400^{th}]$s, $[400^{th}, 600^{th}]$s, $[600^{th}, 800^{th}]$s, $[800^{th}, 1000^{th}]$s. Figure 4.16 is the result. Both simulation results show that AVQ is especially sensitive to changes in link capacity. When C is above 20Mbps, AVQ loses control over queue length. In both scenarios, the slow response of PI leads to large queue delay in at the first 200s and a period of the underutilization after 200th second. REM, LRED and Q-SAPI all effectively control queue length. But the excessive control of REM over queue length results in lower link utilization than LRED and Q-SAPI. There exists tradeoff between queue delay and link utilization. After the 500th s, PI leads to high link utilization by having queue fluctuations larger than REM, LRED and Q-SAPI.
Figure 4.15 Dynamic capacity in [10Mbps, 50Mbps]
4.4.4 Multiple links

This simulation investigates the behavior of each scheme in the network scenario with 5 bottleneck links. Figure 4.17 shows the network topology. The simulated network has 5 bottleneck links (link1-2, link2-3, link3-4, link4-5, link5-6), each with capacity of 15
Mbps and a transmission delay of 20 ms. The edge links are 100 Mbps with 20ms for link delay. Each dark arrow in Figure 4.17 represents 50 FTP flows plus 150 Web sessions, and each light arrow represents 50 backward FTP flows. Another 100 FTP flows go through the left hand side to the right hand side. Other settings are as default.

Figure 4.18 depicts queue length variation of 3 queues versus time under each scheme. Figure 4.19 depicts each bottleneck link utilization variation versus time under each scheme. The results confirm the above discussions.

4.5 Summaries

In this chapter we propose a *queue-based* adaptive PI controller for AQM. This controller aims to improve the transient performance of the fixed-gain PI controller while maintaining its steady state performance over a wide range of uncertainties. It achieves these performance goals by using high controller gains to produce a fast response, followed by low gains to prevent excessive queue oscillations. We perform the stability analysis of the closed-loop system and validate the design through extensive simulations. Simulation results display its versatile ability in addressing the tradeoff between responsiveness and small steady state error under the dynamic network and traffic conditions.

Note that the implementation of Q-SAPI in this chapter achieves these goals with larger computational cost, compared to the existing AQM schemes. Future work will quantify this cost and will explore the efficient implementation.
By now, we only investigate Q-SAPI in the simulator. Examining it in real networks is important. Although we have done simulations in the networks with multiple bottleneck links, we only carry out the analysis of robustness in the situation of a single bottleneck link. In addition, we assume homogeneous delay in analyzing system stability. Future research is considering these problems.
Figure 4.19  Bottleneck Link Utilization variation over time (second)
CHAPTER 5

THE EFFECTS OF AQM ON THE PERFORMANCE OF ASSURED FORWARDING SERVICE

5.1 Introduction

The current AF-based service framework employs RIO [9] mechanism as AQM. In Chapter 4, we have mentioned that much research effort in the Internet community has been devoted to designing robust and efficient AQMs due to the weakness of RED as AQM. However, most of these new AQMs are evaluated only in best-effort networks. In addition, most of these intelligent traffic conditioners, discussed in Chapter 3, are evaluated only in the network scenarios where RIO is employed as AQM. To the best of our knowledge, no work has been carried out to investigate the abilities of intelligent traffic conditioners under different AQM schemes.

The aforementioned points serve the motivation for the work in this chapter. This chapter presents an empirical study of the effects of AQMs on the performance of the AF service in the non-oversubscribed DiffServ networks. An oversubscribed link refers to the link where the sum of CIRs of all competing aggregates is less than the link capacity. The AQMs to be studied include RED, Adaptive RED (ARED), fixed-gain PI controller [32], Adaptive Virtual Queue (AVQ) [46] below, Random Exponential Marking (REM) [4], Loss Ratio Based RED (LRED) [78], and a new self-tuning AVQ (STAVQ) mechanism, which is proposed in this chapter. The reasons for choosing the AQM schemes are: (i) they are implemented in ns-2 or their authors give the detailed implementation and parameters settings; (ii) these AQMs are representative of improved RED, rate-based AQM, queue-based AQM, and self-tuning AQM. Rate-based AQMs use the arrival rate to measure congestion but queue-based AQMs use the queue length to measure congestion.

PI controller is an example of queue-based AQM and AVQ is an example of rate-based. In this chapter we use the implementation of REM in ns-2, which uses queue length and arriving rate to measure congestion. The performance metrics under consideration include link throughput and the achievement of bandwidth assurance and the attainment of
excessive bandwidth. The standard traffic conditioner mentioned in this chapter is TSW2CM. Note that we only study these AQMs in the case that there are two priority levels (IN and OUT) and parameters of each AQM scheme are set in the non-overlapping way. However, the results obtained can be extended to the case where there are three priority levels. By non-overlapping, we mean:

\[
\begin{align*}
    p_{\text{out}} < 1 & \Rightarrow p_{\text{in}} = 0 \\
    p_{\text{in}} > 0 & \Rightarrow p_{\text{out}} = 1
\end{align*}
\]

where \( p_{\text{in}} \) and \( p_{\text{out}} \) are the dropping probabilities of IN and OUT packet, respectively.

The rest of this chapter is organized as follows. First, some related work is given in Section 5.2. Section 5.3 presents the implementations of PI, REM, AVQ and LRED in the two-priority-level form. Section 5.4 gives simulation configurations. Section 5.5 investigates the existing AQMs when the standard traffic conditioner is used. Section 5.6 investigates the effects of the existing AQMs on the ability of an intelligent traffic conditioner. In Section 5.7, some discussions are first given. Then STAVQ is presented and evaluated. We give conclusions in Section 5.8.

### 5.2 Related work

In order to improve bandwidth assurance and fair share in excessive bandwidth in the current AF-based DiffServ framework, some researchers such as in [11] designed intelligent AQM schemes with the assumption that core routers have certain information of flows or aggregates. Consistent with the DiffServ Concept that complexity is pushed to the edges of the networks in order to make this framework scalable, flexible and easy to be introduced into the Internet, the less the information maintained at core routers, the better. Thus we only consider those AQMs, aimed for facilitating congestion control. That is, core routers don’t maintain per-flow/per-aggregate information and differentiate packets based solely on the DSCP marking of the packets.

The effects of RIO on the performance of the AF service when the standard traffic conditioner is used have been studied in the past years. The authors in [56] modeled the DiffServ architecture and found that “choice of different RIO parameter values can have a clear impact on system performance”. Similar work is carried out in [57]. The authors in [57] point out that setting the loss probability threshold of OUT packets needs to make a
tradeoff between network utilization and the fairness among TCP connections. The authors in [59] investigated the effects of different implementations of multilevel RED on the performance of the AF service. They recommended the “staggered” model as best suited to achieve the performance in the AF PHB.

The closest work to ours is the one presented in [76], which studies the effect of a rate-based AQM on the performance of the AF service and shows that a rate-based AQM produced better performance than a queue-based AQM via simulations. In order to make the rate-based AQM work when more than one service class shares the same physical link capacity, they also propose a method of allocating the link bandwidth among these classes. They only investigate the impact of AQM on the system performance when the standard traffic conditioner is used. In addition, they only consider the fixed-gain AQM controller.

In order to alleviate the failure of providing bandwidth assurance in the AF-based DiffServ networks, some intelligent traffic conditioners (see Chapter 3) have been developed at ingress routers. Most of these intelligent traffic conditioners are evaluated only in the network scenarios where RIO is employed as AQM. Although these intelligent conditioners significantly improve bandwidth assurance, they fail in some non-oversubscribed scenarios. In Chapter 3 we systematically analyze some intelligent conditioners and concluded that the design issue leads to the failure. We also derive the conditions, which guarantee the achievement of bandwidth assurance in the network scenario, where network parameters are known. Although we only consider the network situations that RIO is used as AQM, the derived conditions suggest the importance of maintaining the stable queue length on guaranteeing bandwidth assurance. The authors in [14] investigate the ability of an intelligent traffic conditioner when a two-level Proportional-Integral (PI) controller is used as AQM. They do not compare the behaviors of difference AQMs.

5.3 The implementations of some two-level AQMs

In this section, we present the implementations of PI controller, AVQ, REM, and LRED in the two-priority-level form with non-overlapping marking. The common variables used in all these mechanisms are listed in Table V.
Table V  Common Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{curQ}_{\text{in}} )</td>
<td>number of IN packets in the current buffer</td>
</tr>
<tr>
<td>( \text{curQ}_{\text{out}} )</td>
<td>number of OUT packets in the current buffer</td>
</tr>
<tr>
<td>( \text{oldQ}_{\text{in}} ) instant</td>
<td>number of IN packets in the buffer at the last time</td>
</tr>
<tr>
<td>( \text{oldQ}_{\text{out}} ) instant</td>
<td>number of OUT packets in the buffer at the last time</td>
</tr>
<tr>
<td>( \text{p}_{\text{in}} )</td>
<td>dropping probability of IN traffic</td>
</tr>
<tr>
<td>( \text{p}_{\text{out}} )</td>
<td>dropping probability of OUT traffic</td>
</tr>
<tr>
<td>( \text{curV}_{\text{in}} )</td>
<td>current arrival rate of IN packets</td>
</tr>
<tr>
<td>( \text{curV} )</td>
<td>current arrival rate of all packets</td>
</tr>
<tr>
<td>( q_{\text{max}} )</td>
<td>the maximum queue length above which all arriving OUT packets are dropped</td>
</tr>
<tr>
<td>( q_{\text{buf}} )</td>
<td>the target number of IN packets in the buffer</td>
</tr>
<tr>
<td>( q_{\text{out}} )</td>
<td>the target queue length</td>
</tr>
<tr>
<td>( q_{\text{lim}} )</td>
<td>buffer size</td>
</tr>
<tr>
<td>( C )</td>
<td>link bandwidth</td>
</tr>
</tbody>
</table>

5.3.1 Two-level PI controller

PI Controller is a queue-based AQM mechanism, which uses queue length and the changing rate of queue length to measure congestion. The algorithm of two-level PI is given in Figure 5.1. Now we give some explanations of the implementation. Because the dropping probability of OUT packets in PI is accumulated step by step, buffer overflow may occur during this accumulation process, resulting in the undesired IN packet drops and then resulting in the undesired failure of bandwidth assurance. Thus, in Figure 5.1 as long as the queue length exceeds a certain value defined as \( q_{\text{OUT}}^{\text{max}} \), all the arriving OUT packets are dropped even when ECN-marking is used to notify of incipient congestion when the queue length is below \( q_{\text{OUT}}^{\text{max}} \). This rule is also used in AVQ, REM and LRED. The value of \( q_{\text{OUT}}^{\text{max}} \) in PI should be large in order to avoid the slow accumulation of the dropping probability of OUT packets. But for AVQ and REM and LRED, \( q_{\text{OUT}}^{\text{max}} \) can be small. The reason is that the arriving rate participates in measuring congestion in AVQ and REM; the loss ratio estimated periodically is used as the steady state dropping probability in LRED.
Variables

\( a \) and \( b \) are user-defined positive parameters

Periodically

\[
\begin{align*}
    p_{in}(k) &= p_{in}(k-1) + a(curQ_{in} - q_{in}^{ref}) + b(oldQ_{in} - q_{in}^{ref}) \\
    p_{out}(k) &= p_{out}(k-1) + a(curQ_{out} + curQ_{out} - q_{ref}) + b(oldQ_{in} + oldQ_{out} - q_{ref})
\end{align*}
\]

For each arrival packet

If overflow, drop packet

If IN packet and  \( p_{out}(k) \leq 1.0 \), randomly drop packet

If OUT packet and  \( (curQ_{in} + curQ_{out}) > q_{out}^{max} \), drop packet

else randomly drop packet

---

5.3.2 Other two-level AQMs

Figure 5.2 and Figure 5.3 describe two-level algorithms of AVQ and REM, respectively. The algorithm for two-level LRED is given in Figure 5.4. In Figure 5.4, the loss ratio of OUT packets is computed based only on OUT packets, but the current queue length is used to compute the dropping probability of OUT packets. That is, all packets in the buffer are considered. Unless otherwise specified, in the following AVQ, REM, and LRED denote their two-level forms, respectively.

5.4 Simulation configuration

We use ns-2 to investigate RIO, ARIO, PI, AVQ, REM, and LRED in different network scenarios. ARIO denotes two-level ARED. A network scenario is defined as a combination of a traffic mix and a network topology.

The dumbbell network topology for simulations is depicted in Figure 5.5, where 10 aggregates (A_1-A_{10}) share a single bottleneck link (C_1—E_1). In this figure S_i/D_i (1\leq i \leq 10) is source/destination node; I_1 is ingress router; E_1 is egress router; and C_1 is core router. The link delay of link C_1—E_1 is 10ms. The capacities and delays of other links are 100Mbps and 5ms, respectively. The flows of aggregate A_i go from S_i to D_i. In default, all connections have the same round-trip link delay, 50ms. A_1-A_7 are adaptive aggregates, each consisting of 5 identical FTP connections (micro-flows); A_8 sends out HTTP traffic, variable with Pareto-II distribution. The shape parameter is set to 0.97 and the scale

93
Variables

$\alpha$ and $\gamma$ are user-defined parameters, $b$ is packet size

For each arrival packet

$vQ = \max \left\{ vQ - \tilde{C} \times (t_i - t_{i-1}), 0 \right\}

vQ_{in} = \max \left\{ vQ_{in} - \tilde{C} \times (t_i - t_{i-1}), 0 \right\}

If overflow, drop packet

If IN packet

if $vQ_{in} + b > q_{lim} - 1$ or $(curQ_{in} + curQ_{out}) > q_{out}^{in}$, drop packet

else $vQ_{in} = vQ_{in} + b$ and $vQ = vQ + b$

If OUT packet

If $(curQ_{in} + curQ_{out}) > q_{out}^{in}$, drop packet

else if $vQ + b > q_{lim}$, drop packet

else $vQ = vQ + b$

$\tilde{C}(t_i) = \tilde{C}(t_{i-1}) + \alpha \times \left[ \gamma \times C \times (t_i - t_{i-1}) - b \right]$

Figure 5.2 Two-level AVQ

Variables

$\alpha$ and $\gamma$ are user-defined parameters

Periodically

$p_\alpha(k) = p_\alpha(k-1) + \alpha \times \left[ curV_\alpha - C - \gamma \times (curQ_\alpha - q_{\alpha}) \right]

p_{\alpha}(k) = p_{\alpha}(k-1) + \alpha \times \left[ curV_\alpha - C - \gamma \times (curQ_\alpha + curQ_{\alpha} - q_{\alpha}) \right]

For each arrival packet

If overflow, drop packet

If IN packet and $p_{\alpha}(k) = 1.0$, randomly drop packet

If OUT packet, if $(curQ_{in} + curQ_{out}) > q_{out}^{in}$, drop packet

else randomly drop packet

Figure 5.3 Two-level REM

parameter is set to 838 according to [34]. FTP and HTTP connections run over the TCP Reno implementation. $A_8$ and $A_{10}$ sending best-effort traffic. $A_1$-$A_7$ and $A_9$ are AF aggregates and CIR is set to 2Mbps in default. The packet size is set to 1000bytes.
Variables
arrPktNumIN (arrPktNumOUT): number of IN (OUT) packets arriving in the time interval

dropPktNumIN (dropPktNumOUT): number of IN (OUT) packets dropped in the time interval

Periodically

dropNumIN[index] = dropPktNumIN

arrNumIN[index] = arrPktNumIN

for (i=0;i<4;i++) allDropNumIN = allDropNumIN + dropNumIN[i]

for (i=0;i<4;i++) allArrNumIN = allArrNumIN + arrNumIN[i]

lossRatioIN = lossRatioIN × 0.1 + allDropNumIN / allArrNumIN × 0.9

dropNumOUT[index] = dropPktNumOUT

arrNumOUT[index] = arrPktNumOUT

for (i=0;i<4;i++) allDropNumOUT = allDropNumOUT + dropNumOUT[i]

for (i=0;i<4;i++) allArrNumOUT = allArrNumOUT + arrNumOUT[i]

lossRatioOUT = lossRatioOUT × 0.1 + allDropNumOUT / allArrNumOUT × 0.9

index++

For each arriving packet

\[ p_{in} = \text{lossRatioIN} + 0.001 \times \sqrt{\text{lossRatioIN} \times (\text{cur}Q_{in} - q_{ref}^\text{in})} \]

\[ p_{out} = \text{lossRatioOUT} + 0.001 \times \sqrt{\text{lossRatioOUT} \times (\text{cur}Q_{in} + \text{cur}Q_{out} - q_{ref}^\text{out})} \]

If overflow, drop packet

If IN packet and \( p_{out}(k) = 1.0 \), randomly drop packet

If OUT packet, if \( \text{cur}Q_{in} + \text{cur}Q_{out} > q_{out}^\text{max} \), drop packet

else randomly drop packet

Figure 5.4 Two-level LRED

In addition, we consider the network topology with multiple bottleneck links depicted in Figure 5.6. All link delays are set to 5ms. Access link capacity is set to 100Mbps. \( I_k (E_k) \) represents the ingress (egress) router for aggregate \( A_k \) (k=1...14). The characteristics of \( A_{11}-A_{10} \) are the same as in Figure 5.5. The characteristics of \( A_{11}-A_{14} \) are set to the same as \( A_1 \), except that CIR of \( A_{11}-A_{14} \) is set to 0.5Mbps. In all the following simulations, the buffer size of all routers is set to 900 packets. \( q_{out}^\text{max} = 800 \) packets for REM, LRED and PI. But for AVQ, \( q_{out}^\text{max} = 200 \) packets. \( q_{ref}^\text{in} \) is set to 300 packets and \( q_{ref} \) is set to 150 packets.
Figure 5.5  Dumbbell network topology

Figure 5.6  Network topology with multiple bottleneck links

Adaptive hosts and network routers are ECN-enabled. The packet size at routers is 1000 bytes.

Some notations and the corresponding parameters used in the following are defined in Table VI. For each AQM mechanism, we select the parameters in order to let the scheme perform well in most of the following simulations. Unless otherwise specified, the settings in this section are used as default values in the following simulations.

The performance metrics under consideration for investigating steady state behaviors include Average Link Throughput and Average Aggregate Goodput of A1-A7. Average Link Throughput is computed by measuring the number of packets sent out from the router C1 over a specified time period after the network is in the quasi-stable state. Average Aggregate Goodput of an aggregate is computed by measuring the number of packets received at the receivers belonging to this aggregate over a specified time period after the network is in the quasi-stable state. Each simulation is repeated 10 times, and then a final average is taken over all the runs. The performance metrics for investigating transient behavior is Average Goodput Variation versus time and Link Throughput Variation in the simulation period. Both are estimated per 5.0 seconds.
Table VI  Schemes

<table>
<thead>
<tr>
<th>Mechanism</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>RIO</td>
<td>([q_{\min}, q_{\max}, p_{\max}]) for IN and OUT: [200,400, 0.02], [100,200, 0.1]</td>
</tr>
<tr>
<td>ARIO</td>
<td>([q_{\min}, q_{\max}, p_{\max}]) for IN and OUT: [200,400, 0.02], [100,200, 0.1], top=1.0</td>
</tr>
<tr>
<td>AVQ</td>
<td>(\alpha=0.15, \gamma=0.96)</td>
</tr>
<tr>
<td>REM</td>
<td>(\gamma=0.1, \phi=1.001, \alpha=0.01, T=1/160) s</td>
</tr>
<tr>
<td>LRED</td>
<td>(q_{\text{ref}}=150, T=1) s</td>
</tr>
<tr>
<td>PI</td>
<td>(q_{\text{ref}}=150, a=1.822 \times 10^{-6}, b=1.816 \times 10^{-6}, T=1/160) s</td>
</tr>
</tbody>
</table>

5.5 Performance evaluation under standard traffic conditioner

This section investigates the effects of the transient and steady state behaviors of RIO, ARIO, PI, AVQ, REM and LRED when ingress routers employ the standard traffic conditioner.

5.5.1 Dumbbell network topology: effects on the attainment of excessive bandwidth

This subsection investigates the effects of each AQM scheme on the attainment of excessive bandwidth of each aggregate. CIRs of A_1-A_7 are set to 0.4, 0.6, 0.8, 1.0, 1.5, 2.0, and 2.5Mbps, respectively. CIR of A_9 is 0.5Mbps. The sending rates of A_9 and A_10 are both set to 1.0Mbps.

We do simulations in three network scenarios: (i) \(q_{\text{ref}}=50\) packets, \#=30; (ii) \(q_{\text{ref}}=150\) packets, \#=30; (iii) \(q_{\text{ref}}=150\) packets, \#=5. \# denotes the number of micro-flows in each of A_1-A_7. When \(q_{\text{ref}}=50\) packets, \([q_{\min}, q_{\max}]\) for OUT traffic is set to [20, 80] packets in RIO and ARIO. When \(q_{\text{ref}}=150\) packets, \([q_{\min}, q_{\max}]\) for OUT traffic is set to [100, 200] packets in RIO and ARIO. Other settings are set as in Section 5.4. Each simulation lasts 400s. Figure 5.7 depicts the Average Excessive Goodput of each aggregate under each scheme in the three network scenarios. We can see that (i) under each scheme each aggregate can achieve its bandwidth assurance after the network is in the quasi-stable state. (ii) It is not always true that the aggregate with larger CIR obtains larger excessive bandwidth. The number of micro-flows also affects the obtained amount of excessive bandwidth.
bandwidth. Thus, we can see that in the situation of $q_{ref} = 150$ packets and $\# = 30$, aggregate with larger CIR obtains larger excessive bandwidth. But the result is to the contrary in the scenario with $q_{ref} = 150$ packets and $\# = 5$. (iii) When $q_{ref}$ decreases, the difference of excessive bandwidth among $A_1$-$A_7$ increases. That is, the queue length affects the distribution of excessive bandwidth among competing aggregates.

Without loss of generality, we use the results of the scenario of $q_{ref} = 150$ packets and $\# = 30$ to investigate the relations between the queue length and the distribution of excessive bandwidth. Figure 5.8 depicts the queue length variation versus time under each scheme. Average Goodput Variations of $A_6$ and $A_{10}$ under each scheme are depicted in Figure 5.9 and Figure 5.10, respectively. From Figure 5.8, Figure 5.9, and Figure 5.10, we can see that (i) the smaller the queue length, the smaller the goodput of $A_{10}$. The reason is that the smaller the queue length, the larger the dropping probability of OUT packets

98
when the traffic load is unchanged. Then, there are more packet drops from $A_{10}$. The free link bandwidth can be used by AF adaptive aggregates. Thus, the smaller the queue length, the larger the goodput of $A_6$ is, as shown in Figure 5.9. (ii) The queue length under RIO oscillates around the maximum queue threshold of OUT packets (i.e. 200 packets in Figure 5.8). It is due to the incorrect setting of the loss probability threshold of OUT packets.
(iii) Although ARIO, REM, PI, LRED have similar results in Figure 5.7, each aggregate experiences takes different time to reach the value in Figure 5.7 under different AQM schemes.

5.5.2 Dumbbell network topology: effects on the achievement of bandwidth assurance

This section investigates the effects of each scheme on the achievement of bandwidth assurance in two situations: one is that A1-A7 have different CIRs; the other is that their round-trip link delays are different. Each simulation is first done when $q_{ref} = 50$ packets and then it is repeated when $q_{ref} = 150$ packets.

5.5.3 Different CIRs

The target rates of A1-A7 are set to 0.5, 1.0, 1.5, 2.0, 2.5, 3.5 and 5.0Mbps, respectively. Other settings are same as default. Figure 5.11 depicts the Average Goodput Deviation of A1-A7 under each scheme. Average Goodput Deviation is defined as $[(\text{Average Goodput}) - \text{CIR}]$. Table VII gives the link throughput under each scheme. The label “LRED-50” denotes the results of LRED when $q_{ref} = 50$ packets. The label “RIO(20,80)” denotes the results of RIO when $[q_{min}, q_{max}]$ for OUT packets is set to [20, 80]. Definitions of other labels are similar.
Figure 5.11  Average Goodput Deviation of A1-A7 with different CIRs

<table>
<thead>
<tr>
<th>Scheme</th>
<th>RIO (20,80)</th>
<th>RIO (100,200)</th>
<th>ARIO (20,80)</th>
<th>ARIO (100,200)</th>
<th>AVQ</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Scheme</th>
<th>REM</th>
<th>LRED-50</th>
<th>LRED-150</th>
<th>PI-50</th>
<th>PI-150</th>
</tr>
</thead>
</table>

The results show that (i) Rate-based AQM schemes perform better than queue-based AQM schemes in terms of improving bandwidth assurance. (ii) When AVQ and REM are employed, there is no failure of bandwidth assurance. This is achieved by excessively ECN-marking OUT packets, resulting in low link throughput. Although \( q_{ref} \) is set to 150packets in REM, the queue actually oscillates in [0, 50] packets. (iii) For RIO, ARIO, LRED and PI, bandwidth assurance is improved when the traffic load is unchanged and \( q_{ref} \) decreases. The reason is similar to the discussions in Section 5.5.1. That is, the smaller the queue length, the larger the dropping probability of OUT packets, then the aggregate with goodput below its CIR has more chance to improve its goodput. (iv) The incorrect setting of the loss probability threshold of OUT packets in RIO leads to the queue length oscillating around the maximum queue threshold of OUT packets. This results in low link throughput and prevents RIO from improving bandwidth assurance for A2-A7. The efficiency of ARIO, LRED and PI in controlling
the queue length results in the better provision of bandwidth assurance, compared to RIO, and high link throughput, compared to AVQ and REM.

However, different schemes result in different time to reach the values in Figure 5.11. During the transient phase, bandwidth assurance of some aggregates may not be satisfied although they can in the steady state. How long the failure of bandwidth assurance lasts depends on the transient response of an AQM scheme. Figure 5.12 and Figure 5.13 illustrates this. Figure 5.12 depicts the queue length variation under REM and PI-150; Figure 5.13 depicts Average Goodput Variation of A5 under REM and PI-150. The figures show that during the first 90 seconds the average goodput of A5 is below its CIR (2.5Mbps) either under REM or under PI-150. However, REM has quicker transient response than PI-150 and then the queue length is under control more quickly when REM
is used. The right figure shows that A₃ under REM reaches CIR after about the 90ˢ. But under PI-150, it waits until about the 140ˢ.

5.5.4 Different propagation delays

We set round-trip link delays of A₁-A₇ to different values by setting the link delay of E₁—D₁ (i from 1 to 7) to 10, 50, 200, 350, 500, 650, and 800ms, respectively. Other settings are same as in Section 5.4. Figure 5.14 depicts the Average Goodput Deviation of A₁-A₇ under each scheme. Table VIII gives the link throughput under each scheme.

The results show that (i) Different from Section 5.5.3, there is no improvement of bandwidth assurance under RIO, ARIO, LRED and PI when qₖₑₑ decreases. Even worse, the bottleneck link throughput decreases. The main reason is that the AQMs lose control over the queue length. If the queue length is controlled efficiently, dropping a sequence of OUT packet from a TCP connection seldom occurs and then timeout seldom occurs and then A₁-A₇ can start increasing congestion window (cwnd) earlier after cwnd is halved due to congestion. (ii) The better achievement of bandwidth assurance under LRED, PI and REM than ARIO(100,200) is due to the inappropriate settings for OUT packets in ARIO(100,200). We also do simulation with ARIO(50,250). There is some improvement in bandwidth assurance.

5.5.5 Multiple bottleneck links: effect on achievement of bandwidth assurance

This section uses the network topology in Figure 5.6 to investigate the effects of the steady state behaviors of each scheme on the achievement of bandwidth assurance when there are multiple bottleneck links. The target rates of A₁-A₇ are set to 0.5, 1.0, 1.5, 2.0, 2.5, 3.5 and 5.0Mbps, respectively. Other settings are set as in Section 5.4. Figure 5.15 shows the Average Goodput of A₁-A₇ under each AQM scheme when the standard traffic conditioner is used at ingress routers. The results confirm the discussions in Section 5.5.3. CIR of each of A₁₁-A₁₄ is set to 0.5Mbps.
5.6 Performance evaluation under intelligent traffic conditioner

This section investigates the effects of RIO, ARIO, PI, AVQ, REM and LRED on the ability of intelligent traffic conditioners in improving bandwidth assurance. We use as an example of intelligent traffic conditioners the mechanism, namely Variable-Structure Adaptive CIR Threshold (VS-ACT) developed in Chapter 4. The marking threshold

104
(CIR_{Thresh}) of a priority level for an aggregate is the marking rate, determining the allowed maximum low-pass filtered arrival rate of the packets of this aggregate belonging to the priority levels no lower than this priority level. We only consider the scenarios of a single bottleneck link.

We do simulations in two network scenarios: different RTTs and different numbers of micro-flows in an aggregate. The first scenario aims to investigate the achievement of bandwidth assurance and link throughput under different AQM schemes. The second scenario aims to investigate the importance of low bound of CIR_{Thresh} in intelligent conditioners and to investigate the reasons.

### 5.6.1 Simulation 1: in the network scenario of Section 5.5.4 (Different RTTs)

We use the network scenario in Section 5.5.4 but employ the intelligent conditioner VS-ACT at ingress routers. Figure 5.16 shows the Average Goodput Deviation of A_1-A_7 for each scheme. Table IX gives the link throughput under each scheme. The results show that ARIO(100,200), ARIO(20,80) and LRED-150 performs best in term of link throughput and the achievement of bandwidth assurance. This is because they can control queue length in a range above zero. Other schemes can cooperate with VS-ACT to alleviate the failure of bandwidth assurance. However, they either lose control over queue length such as PI and RIO, or excessively control queue length such as AVQ and REM, resulting in excessive congestion notifications to the sources. Although PI and RIO both lose control over queue length, the results are different. The queue length under PI-150 fluctuates seriously and experiences zero length periodically. But the queue length under RIO fluctuates around $q_{OUT}^{max}$. Thus, the link throughput under PI and AVQ and REM decreases, compared to in Section 5.5.4. But RIO and ARIO under intelligent conditioners increase link throughput.
5.6.2 Simulation 2: Different # of micro-flows in each aggregate

This section aims to show the importance of efficiently controlling queue length in the scenario that \( A_1 - A_7 \) have different number of micro-flows. In addition, we aim to show the importance of bounding \( CIR_{thresh} \) from below.

The simulation results in Section 5.6.1 illustrate that VS-ACT with lower bound efficiently improves bandwidth assurance under all these schemes. In this section, we show that in some network scenarios, when an AQM scheme degrades to Drop-tail-OUT, an intelligent conditioner which does not use CIR as the lower bound of \( CIR_{thresh} \) degrades the achievement of bandwidth assurance rather than improves, compared to using the standard traffic conditioner. By Drop-tail-OUT, we mean that the queue length oscillates around \( q_{OUT}^{max} \).

Due to the slow transient response of REM and PI, in the following we do simulations only with RIO, ARIO, LRED and AVQ when the standard traffic conditioner, VS-ACT without lower bound (VS-ACT-NB) and VS-ACT with lower bound (VS-ACT-B) are
applied, respectively. By with lower bound, we mean that $CIR_{\text{_thresh}}$ is not less than CIR. Note that AVQ performance is sensitive to the value of $\gamma$. We use AVQ98 to denote AVQ using $\gamma = 98\%$. Similar definition is for AVQ96. We use the default settings of each aggregate in Section 4 except that the number of micro-flows of A1-A7 is set to 5, 10, 15, 20, 25, 30, 35, respectively.

The simulation results show that when ARIO, LRED and AVQ96 are employed, all aggregates can achieve their CIRs no matter whether VS-ACT-NB or VS-ACT-B is used. We omit their results. But these are not for RIO and AVQ98. Figure 5.17 gives the average goodput of A1-A7 in each scenario where RIO or AVQ98 is used as AQM, and standard conditioner or VS-ACT-B or VS-ACT-NB is employed at ingress routers. We can see that when RIO and VS-ACT-NB is used, A1-A7 fail to achieve bandwidth assurance. The main reason is that the intelligent conditioner assumes that the average arriving rate of an aggregate at ingress routers can represent the average goodput of this aggregate. But this assumption is incorrect when an AQM scheme degrades to Drop-tail-OUT. Then the intelligent conditioner makes the wrong decision. This is illustrated in Figure 5.18, which depicts the average arriving rate variation, $CIR_{\text{thresh}}$ variation and average goodput variation of A7 when VS-ACT-NB and AVQ98 are employed.

Note that, in some heavy-subscribed network scenarios, this degradation is unavoidable no matter what AQM scheme is used. Is it possible to prevent the performance degradation caused by employing edge-based mechanisms? The results of using VS-ACT-B in Figure 5.17 give a solution. That is using CIR as the lower bound of $CIR_{\text{thresh}}$.

### 5.7 STAVQ

In this section, we first give some comments based on the above simulation results. Then we present an improved AVQ, namely, self-tuning AVQ (STAVQ) aimed for compensating the insufficiency of AVQ.
Figure 5.17 Average Goodput of A1-A7

Figure 5.18 A7 under VS-ACT-NB and AVQ98

5.7.1 Discussions

It is obvious that the smaller steady state queue length is helpful to improve bandwidth assurance in the heavy-subscribed network scenarios for queue-based AQMs. But too small queue length may lead to low link throughput. RIO is sensitive to traffic load. Its insufficiency in controlling queue length results in the failure of bandwidth assurance in some scenarios where there is no such failure when other AQMs are used. Due to the fixed controller gains, PI and REM often show slow transient response and then they are not suitable in the dynamic AF-based DiffServ networks. In all the above simulations, AVQ96 and ARIO and LRED exhibit comparable performance in terms of the achievement of bandwidth assurance, better than other schemes. However, the ability of LRED depends on whether it could estimates the loss ratio. The wrong estimation can
degrade link throughput. Completely depending on queue length to control congestion, ARIO display slow transient response in some scenarios and then affects the achievement of bandwidth assurance.

Using rate to measure congestion can produce fast transient response. However, there is a problem with AVQ. That is, the performance of AVQ depends on the assumption that $C$ is known. All the above simulations are done based on the assumption that there is only one physical queue at the router. In practice, however, a single physical link is often divided into several virtual links, each for a physical queue. Each physical queue is used for a service class. The fixed assignment of link capacity degrades the physical link throughput. The assumption of a fixed capacity assigned to a virtual link (a physical queue) is often violated in reality. This problem exists in any AQM scheme when the physical queue service rate is used in computing dropping probability. The authors in [74] proposed a solution to this problem. No matter how this problem is solved, the service rate of a queue is varying in order to utilize the link capacity efficiently. However, it is impossible for AVQ with a certain $\gamma$ to produce good performance in different service rates. The inappropriate $\gamma$ either results in a slow speed in controlling queue length or results in the queue length around $q_{\text{OUT}}^{\text{max}}$ and then affects the achievement of bandwidth assurance and link throughput. This motivates us to improve AVQ.

5.7.2 STAVQ mechanism

STAVQ is an improved AVQ. It aims to alleviate the weakness of AVQ while maintaining its fast transient response.

This section describes the self-tuning AVQ scheme. Figure 5.19 describes this AQM scheme. The main idea is that when the average queue length is larger than $q_{\text{ref}}$, $\gamma$ is decreased and then dropping or ECN-marking probability increases; otherwise increases. The motivation of the algorithm of adjusting $\gamma$ in Figure 5.19 is the steady state
function of dropping probability in [46]. In that function, dropping probability is decreasing function of $\gamma$. $q_{ref}$ in Figure 5.19 is a reference to adjust $\gamma$. It doesn’t mean that the steady state queue length under STAVQ must oscillate around $q_{ref}$. Note that too small $q_{ref}$ may result in low link throughput. In Section 5.7.3, we illustrate that STAVQ avoid the degradation of AVQ98 to Drop-tail-OUT. In Section 5.7.4, we compare STAVQ, ARIO, LRED and AVQ98 in dynamic networks.
5.7.3 Using network scenario in Section 5.6.2

This simulation aims to show the improvement of STAVQ over AVQ98 in terms of bandwidth assurance. The initial $\gamma$ is set to 99%. We repeat the Section 5.6.2 simulations with STAVQ under $q_{ref} = 50$ packets and $d_{ref} = 100$ packets. STAVQ50 denotes STAVQ using $q_{ref} = 50$ packets. Figure 5.20 depicts the queue length variation. It shows that queue length fluctuates a bit above $d_{ref}$ under STAVQ100. Average Goodput Variation of each aggregate under STAVQ50 is similar to under STAVQ100. Here we only show the average arriving rate variation, $CIR_{\text{Thresh}}$ variation and average goodput variation of $A_7$ when VS-ACT-NB and STAVQ50 are employed. The results are in Figure 5.21. We can see that STAVQ improves bandwidth assurance.

5.7.4 Varying UDP load

This section aims to compare LRED-50, ARIO(30,100), AVQ98, STAVQ50 in the dynamic networks. The network topology used is the dumbbell network topology depicted in Figure 5.5. The network is dynamic due to the varying non-adaptive traffic load. The simulation lasts 400s. The settings of each aggregate are same as in Section 5.4 except that the sending rates of $A_9$ and $A_{10}$ are both 0.6 Mbps in [0, 100]s, 5.0Mbps in [100, 200]s, 0.01Mbps in [200,300]s and 9.0Mbps in [300,400]s, respectively.

Figure 5.22 depicts some simulation results. The first row depicts Link Throughput Variation of $C_1-E_1$ versus time under each scheme. The second row depicts Average Goodput Variation of $A_8$ versus time. The third row depicts the Queue Length Variation of $C_1-E_1$ versus time. The results show that: (i) Incorrect estimation in LRED results in the link throughput degradation; (ii) The slow transient response of ARIO to the traffic load changes at the 100th s and the 300th s results in the goodput of $A_8$ below its CIR for a time. (iii) AVQ98 loses control over queue length and then the goodput of $A_8$ is below its CIR in [100th, 200th] and [300th, 400th]. (iv) STAVQ displays fast transient response and then efficiently controls queue length. Thus, the link throughput is high and bandwidth assurance is achieved.
5.8 Summary

In this chapter, the effects of seven AQM schemes in the two-level form on the AF service are investigated. The results show that (i) When the standard traffic conditioner is used, the transient response and the ability of controlling queue length of an AQM scheme affect the achievement of bandwidth assurance, the attainment of excessive bandwidth and link throughput. (ii) The behaviors of an AQM scheme affect the ability of intelligent traffic conditioners. (iii) Self-tuning AQM schemes perform better than AQMs with fixed-gains in terms of the achievement of bandwidth assurance and link throughput. These results can be applied for the interaction of AQMs and intelligent end-to-end mechanisms.
Figure 5.22  dynamic UDP traffic load
CHAPTER 6
IMPROVING APPLICATION LEVEL DOWNLINK AND UPLINK FAIRNESS IN THE INFRASTRUCTURE 802.11 WLAN

6.1 Introduction

Many wireless products do not support the PCF mode [28]. Thus, this chapter restricts the attention to some issues in the infrastructure IEEE 802.11 WLANs using the DCF mode.

The DCF in the legacy IEEE 802.11 provides a channel access with equal probabilities to all the competing nodes, including mobile nodes and the AP. This feature results in the difference between the attained bandwidth of downlink flows and that of uplink flows when there is more than one uplink flow. This difference is increased with the increased number of active mobile nodes. Thus, the MAC-layer fairness does not suggest application level uplink/downlink fairness, which has been pointed out in [47].

Studies of WLAN traffic at university campuses and a multi-day conference show that TCP accounted for more than 90% of bytes exchanged over the WLANs [74]. When there exist TCP downlink and uplink flows, TCP congestion control mechanisms exacerbate this unfairness. In this case, the packets arriving at the AP from the wired networks include data packets of downlink flows and ACK packets of uplink flows. Before the packets are sent to mobile nodes, they are first placed in IFQ, between the logic link (LL) layer and the MAC layer. This queue is managed in the DropTail fashion for default. Currently 100Mbps Fast Ethernet has been widely deployed in a corporate network. The limited channel capacity of the wireless interface at the AP, compared to that of the wired interface, easily leads to IFQ buffer overflow. A data packet drop can lead to the rate halving of a TCP flow. However, this is not true for an ACK packet drop. As long as the ACK packet of at least a data packet in the congestion window arrives at the sender, the congestion window size increases. In some situations, the uplink TCP flows starve the downlink TCP flows. Here we assume that congestion window size can increase infinitely as long as there is no congestion.
This chapter considers two issues under saturated traffic conditions: congestion control at the AP and improving uplink/downlink fairness from the application point of view when non-adaptive flows and adaptive flows coexist. Although some studies have been done to the problem of uplink/downlink unfairness, most of them either consider pure TCP or pure UDP flows or they assume that some knowledge is known, such as the maximum available wireless bandwidth, the number of active flows. In addition, some authors such as in [66] and [79] propose to use local scheduling to achieve flow-based QoS guarantee. Although these proposals can easily be tailored for achieving uplink/downlink fairness, they both depend on the specific MAC protocol. Can these two issues be attacked independent of the specific MAC protocol? Note that such independence can enhance the feasibility and scalability of the mechanism.

Adjusting the values of inter-frame space (IFS) time interval or backoff time interval of DCF is a possible approach. However this kind approach needs to make a tradeoff. Modifying the parameters of mobile nodes requires the cooperation from mobile nodes below the IP layer. In addition, RTT of a TCP connection is increased with the increasing number of mobile nodes, resulting in the large congestion window size and much more traffic burst. TCP combined with RED is unstable when delay is increased [52]. Modifying the parameters of AP avoids the cooperation but may decrease the wireless bandwidth utilization.

ECN-marking is known as an effective mechanism that can be used with AQMs to control congestion in wired networks. In addition, this technique has already been applied to improve the wireless network performance. Some researchers propose to extend the use of ECN to design a loss discriminator in the WLANs, i.e., a technique to distinguish congestion losses from wireless losses. TCP is still the de facto standard for reliable data communications and many research efforts have been devoted to designing AQM schemes in the past years. We are not aware of any extensive work about using ECN-marking for congestion control at the AP when there are both uplink and downlink TCP flows.

These discussions motivate us to explore whether we can extend the existing AQM schemes to handle congestion at AP and improve uplink/downlink fairness. In the following of this chapter, we first review the problem of the uplink/downlink unfairness.
in Section 6.2. We present our approach, namely AQM$^\text{smac}$, in Section 6.3. AQM$^\text{smac}$ does not care what transport protocol the flows use. However it assumes that an arriving DATA packet can be distinguished from an ACK packet according to the information in the packet header. Note that ACK in this chapter denotes ACK in the transport protocol. AQM$^\text{smac}$ mainly depends on (i) allocating the IFQ buffer between downlink and uplink flows and (ii) controlling the contention window size of AP to achieve this fairness. All the knowledge used and the remedies initiated are local. We present a performance evaluation in Section 6.4. Conclusions are made in Section 6.5.

6.2 Problem overview

This section conducts simulations to illustrate uplink/downlink unfairness. The simulator used is ns-2. The network topology is given in Figure 6.1. The wireless LAN interface is assumed to be the bottleneck, and there are no packet losses except those triggered by the IFQ queue management mechanism. We assume an ideal channel without transmission errors or hidden terminal problems, i.e. all stations can always hear all the others. The channel bandwidth in WLAN is set to 11Mbps. Let $n$ denote the number of downlink flows and let $m$ denote the number of uplink flows. The flows from $S_{A1} \ldots S_{An}$ to $D_{A1} \ldots D_{An}$ are downlink flows. The flows from $S_{B1} \ldots S_{Bm}$ to $D_{B1} \ldots D_{Bm}$ are uplink flows. We employ UDP sources sending constant bit rate (CBR) traffic as an example of non-adaptive sources. We use TCP sources generating infinite FTP bulk data as adaptive sources. The TCP sources are based on the TCP-Reno implementation. In order to avoid the impact of flow characteristics such as RTT, we assume all flows are homogeneous by default. IP Packet size is 1500 bytes.

One performance metric is the Average Goodput, calculated by measuring the number of packets at the receiver over a specified time period after the network is in the quasistable state. We repeat each simulation 10 times. The results shown below represent averages from all runs. Another performance metric is the Average Goodput (estimated per 5.0 seconds) variation in the simulation period. In default, the label “Downlink” denotes the results of downlink flows; the label “Uplink” denotes the results of uplink flows; the label “SUM” denotes the results of all competing flows.
6.2.1 Pure UDP+DropTail

All downlink and uplink flows are UDP flows. \( n=m=5 \). The sending rate of each flow is 1Mbps. IFQ queue management scheme is DropTail. We do simulations under different buffer sizes. Figure 6.2 shows the results. We could see that the goodput ratio between uplink flows and downlink flows is about 5:1, consistent with the results of the equal access opportunity among computing nodes when using the legacy DCF.

6.2.2 Pure TCP+ DropTail

This section aims to show that TCP congestion control mechanisms exacerbate the uplink/downlink unfairness when the IFQ buffer is managed by the DropTail scheme. The flows in the same direction are homogeneous. The wired link propagation delay of an uplink flow is 208ms; the link propagation delay of a downlink flow is 12ms. We use these settings in order to show that for uplink flows, even their RTTs are large and the number of flows is small, uplink flows can occupy most of the bandwidth. Figure 6.3 (a) and (b) and (c) depict the goodput variation of both directions under \( n:m=1:1 \), \( n:m=2:1 \) and \( n:m=20:2 \), respectively. Table X gives the average goodput of the last 300 seconds. According to the equal accessing probability among mobile nodes and AP provided by the MAC protocol, the goodput (\( T_{B_{down}} \)) of all downlink flows should at least have been \( 1/(m+n) \) of the goodput (\( T_{B_{up}} \)) of all uplink flows. However Figure 6.3 (b) shows that \( T_{B_{down}} \) in the steady state is less than 1.0Mbps. Figure 6.3 (c) shows that uplink flows nearly starve downlink flows. That is, TCP congestion control mechanisms exacerbate the uplink/downlink unfairness. Note that these results and conclusions are same for any AQM scheme, which drops packets when congestion occurs.

6.3 The proposed approach

In this section we present AQM^{smac}. The description and explanation are given in the following.

6.3.1 Description of AQM^{smac}

This approach consists of three components:
(i) **Rate Estimator (RE).** Deployed at the LL layer, it calculates the arrival rate \(V_{up}\) of data packets of uplink flows and the arrival rate \(V_{down}\) of data packets of downlink flows. Here we assume that RE can distinguish data packets from ACK and other control packets.

(ii) **Adapter.** Also deployed at the LL layer, Adapter uses the algorithm in Figure 6.4 to adjust \(p_{down}\) and \(p_{up}\) and \(CW_{minAP}\). The computed \(p_{down}\) and \(p_{up}\) are sent to \(A^{2}\)RED, and \(CW_{minAP}\) is sent to the MAC layer to update contention window size of AP.

(iii) **IFQ Queue management mechanism: A^{2}RED.** \(A^{2}\)RED uses ECN-marking to notify of congestion. Different from the AQM schemes in the wired networks, \(A^{2}\)RED ECN-marks both Data packets and ACK packets. The processing unit of \(A^{2}\)RED is packet. UDP flows are non-adaptive. We have repeated all the following simulations using randomly dropping UDP flows and using ECN-marking UDP flows. There is less
Figure 6.3 Goodput variation versus time under TCP+DropTail

Table X Average Goodput

<table>
<thead>
<tr>
<th>n:m</th>
<th>1:1</th>
<th>2:1</th>
<th>20:2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Downlink Goodput (Mbps)</td>
<td>0.8329</td>
<td>0.64068</td>
<td>0.08</td>
</tr>
<tr>
<td>Uplink Goodput (Mbps)</td>
<td>2.8287</td>
<td>3.041</td>
<td>4.1548</td>
</tr>
</tbody>
</table>
Variables:
$\alpha, \beta, \gamma$: user-defined variables, $\in (0, 1.0)$

$V_{down}$ ($V_{up}$): the average arriving rate of downlink (uplink) data packets over a sampling period. The unit is Mbps

NumACK: the number of ACK in the last sampling period

Periodically:
if $\{ V_{up} - V_{down} > 0 \}$ then
  if $\{ $AvgRate\_DOWN\_TCP\_ACK > 1.0 \}$ then
    $cwminTemp(k) = cwminTemp(k-1) \times 2$
  else
    if $\{ p_{up} < 1.0 \}$ then
      $p_{up}(k) = p_{up}(k-1) + \alpha (V_{up} - V_{down})/(V_{up} + V_{down})$
    else
      if $\{ p_{down} > 0.0 \}$ then
        $p_{down}(k) = p_{down}(k-1) - \beta (V_{up} - V_{down})/(V_{up} + V_{down})$
      else
        if $\{ $AvgRate\_DOWN\_TCP\_ACK < 1.0 \}$ then
          $cwminTemp(k) = cwminTemp(k-1) - \gamma (V_{up} - V_{down})/(V_{up} + V_{down})$

else
  if $\{cwmin\_BS(k) < 31\}$ then
    $cwminTemp(k) = cwminTemp(k-1) - \gamma (V_{up} - V_{down})/(V_{up} + V_{down})$
  else
    if $\{ p_{down} < 1.0 \}$ then
      $p_{down}(k) = p_{down}(k-1) - \beta (V_{up} - V_{down})/(V_{up} + V_{down})$
    else
      $p_{up}(k) = p_{up}(k-1) + \alpha (V_{up} - V_{down})/(V_{up} + V_{down})$

$p_{down}(k) = \text{max} \{ \text{min} \{ p_{down}(k), 1.0 \}, 0.0 \}$

$p_{up}(k) = \text{max} \{ \text{min} \{ p_{up}(k), 1.0 \}, 0.0 \}$

$CW_{minAP} = \text{int}(cwminTemp)$

$CW_{minAP} = \text{max} \{ \text{min} \{ cwmin\_BS, 1.0 \}, 31.0 \}$

$CW_{maxAP} = 3 \times CW_{minAP}$

Figure 6.4 Algorithm of adjusting $p_{down}$, $p_{up}$ and $CW_{minAP}$
difference on the achieved uplink/downlink fairness. For any arriving packet, A²RED first computes the marking probability $p_{\text{ared}}$ using the standard ARED algorithm. If it is ACK packet, the randomly marking probability is $p_{\text{ared}} \times p_{\text{up}}$; if it is a DATA packet, the randomly marking probability is $p_{\text{ared}} \times p_{\text{down}}$; if it is a control packet, A²RED puts it in the front of IFQ queue and make decision about how to deal with the last packet according to the packet type if there is congestion.

Figure 6.5 depicts the relations among these three components and LL layer and MAC layer.

6.3.2 Motivation and explanation

In this section, we first give the system model of TCP/A²RED in the infrastructure Wireless LAN. Then we explain AQM$^{\text{mac}}$.

6.3.2.1 System model

In wired networks, the capacity of the link connecting two neighbouring nodes is fixed and rarely changes, and each link is dedicated to traffic going in a single direction. On the contrary, in wireless networks, the channel is shared by traffic in all directions. In the particular case of the infrastructure wireless LANs, the channel is shared by both downlink and uplink flows. The total maximum available wireless capacity is time-varying. Thus, not only the network conditions such as capacity but also the traffic condition in the other direction will affect the maximum achievable throughput for flows in one direction. Recall that Figure 4.1 depicts the AIMD/AQM system in the form of a linearized feed back control system. An assumption made in that model is that the bottleneck link capacity $C$ is constant. This assumption is no longer valid in wireless networks as we’ve discussed.

Let $C$ be the overall bandwidth in a WLAN. We assume that there is the equal number of downlink and uplink TCP flows, denoted by $N$. $W_u$ and $W_d$, respectively, denote the congestion window of the uplink TCP flow and the downlink TCP flow. $R$ denotes the round trip time of each flow. Define $k = \frac{\text{ACK\_packet\_size}}{\text{DATA\_packet\_size}}$ . $(W_{d0}, W_{d0}, R_0)$ denotes the equilibrium point.
Ignoring the control message and collision overhead in the MAC protocol, we obtain

\[ \dot{q} = k \frac{N}{R} W_a + \frac{N}{R} W_d - \left( C - k \frac{N}{R} W_a - \frac{N}{R} W_d \right) \]

(6.1)

where

\[ R = \frac{q}{C - k \frac{N}{R} W_a - \frac{N}{R} W_d} + T_p \]

(6.2)

Here \( T_p \) is round trip propagation time. From Eq. 6.2, we obtain

\[ q = (R - T_p) \left( C - k \frac{N}{R} W_a - \frac{N}{R} W_d \right) \]

\[ \frac{\partial R}{\partial q} = \frac{1}{C + \frac{N}{R} \left( \frac{R_0 - T_p}{R_0} \right) - 1 \left( W_{\delta 0} + kW_{\delta 0} \right)} \]

(6.3)

Now we linearize Eq. (6.1) and obtain

\[ \dot{q} = (k + 1) \frac{N}{R} (W_a + W_d) - C \]

\[ \delta \dot{q} = -\frac{N}{R_0^2} \left[ \frac{1}{C + \frac{N}{R_0} \left( \frac{R_0 - T_p}{R_0} \right) - 1 \left( W_{\delta 0} + kW_{\delta 0} \right)} \right] (1 + k)(W_{\delta 0} + W_{\delta 0}) \delta q + (k + 1) \frac{N}{R_0} (\delta W_a + \delta W_d) \]
The interaction between the uplink and downlink traffic dynamics is illustrated in Figure 6.6. From this figure, we can see that the output of the upper block is an input to the downlink block. In order to control the IFQ queue behavior, we can use any queue-based AQM controller that we have investigated in Section 4.

6.3.2.2 The motivation of using dynamic contention window size of base station

When there only exist TCP flows, $A^2$RED can achieve Dowlink_goodput $\approx$ Uplink_goodput. However, when the total sending rate of UDP flows is larger than the maximum available wireless bandwidth, $A^2$RED displays its insufficiency. $A^2$RED needs help from the MAC layer. The insufficiency can be alleviated by dynamically adjusting IFS, backoff interval or contention window size of based station or contention window size of mobile nodes. We have done many simulations and find that dynamically changing contention window size is more effective for improving uplink/downlink fairness. The problem of adjusting contention window size of AP is that in some situation the uplink/downlink fairness can not be achieved, which can be achieved if we adjust contention window size of mobile nodes. But in the following we prefer to adjust the contention window size of AP in order to localize the actions. In addition, we examine how much improvement in uplink/downlink fairness can be achieved.

When the number of competing flows is very large, the collision possibility may increase and the maximum available wireless bandwidth may decrease if we only change contention window size of base station. The following simulations will illustrate these scenarios. Thus, we do not adjust contention window size of AP unless $A^2$RED is not effective.

The reason we adjust $p_{up}$ is as follows. Consider a scenario: downlink flows are UDP; uplink flows are long-lived TCP; the sending rate of UDP flows is larger than the maximum available bandwidth. If we let $p_{up}=1$, we can not achieve the fairness if we do not adjust contention window size of AP.

In summary, the principle in the algorithm in Figure 6.4 is that we adjust contention window size only when $A^2$RED is not effective in order to prevent the decrease in wireless link utilization.
6.3.2.3 The method of adjusting $p_{\text{down}}$, $p_{\text{up}}$, $CW_{\text{minAP}}$

The algorithm in Figure 6.4 uses fixed-gain PI-controller to adjust $p_{\text{down}}$, $p_{\text{up}}$, $CW_{\text{minAP}}$. We have discussed that such controller needs to make a tradeoff between fast transient response and small steady state error. This chapter focuses on investigating the correctness of the algorithm in Figure 6.4. We leave the improvement of the algorithm for future work.

6.4 Performance evaluation

The performance of FAIR and DyCW also are not affected by the used transport protocol. We have ever tried to implement FAIR and done simulations in pure TCP scenarios or pure UDP scenarios. It is difficult for us to produce their results. Thus, we give up the comparison between FAIR and our approach in the following. In this section, we compare AQM\textsuperscript{mac} with ARED, DyCW. Here ARED denotes that IFQ queue management scheme is ARED and the legacy MAC is used. Note that ARED ECN-marks both DATA packets and ACK packets.

When doing simulations with DyCW, that IFQ queue management scheme is DropTail. In ARED, $[\max_{\text{th}}, \min_{\text{th}}]$ is set to $[\text{qlim}\times70\%, \text{qlim}\times30\%]$; $\text{top}=1.0$; other parameter settings are set as in ns-2. $\text{qlim}$ denotes the IFQ buffer size. Since ARED can effectively control queue length around $\text{qlim}\times50\%$, we use $\text{qlim}/2$ as buffer size when we
do simulations with DyCW and use $q_{lim}$ as buffer size for ARED and AQM$^{smac}$. In AQM$^{smac}$, $V_{down}$ and $V_{up}$ are updated per 1.0 second; $\alpha=\beta=\gamma=0.8$. The algorithm in Figure 6.4 is executed per 1.0 second.

6.4.1 Simulation 1: Pure TCP flows

Let $n=m$ and all flows be homogeneous. We do simulations by changing $n$ and the IFQ buffer size. $n$ is changed from 1 to 30. Buffer size is 100, 300 and 700 packets. The performance metric is average goodput. Unless otherwise specified, all the average goodput in this chapter is the average value of 10 run simulations. In each run the simulation lasts 400s; the goodput is computed in the last 200s. Figure 6.7, Figure 6.8, and Figure 6.9 are the results of ARED, DyCW and AQM$^{smac}$, respectively.

Figure 6.7 shows that ARED efficiently controls IFQ queue length and then improves uplink/downlink fairness. The average goodput of downlink flows is improved greatly and the improvement is larger with the increased buffer size.

Figure 6.8 shows that the uplink/downlink fairness can be nearly achieved. However, the WLAN bandwidth utilization is decreased when the number of mobile nodes is increased, compared to ARED and AQM$^{smac}$. The reason is that it is unnecessary to use so large contention window size of mobile nodes. The severe problem of DyCW in this case is that RTT is very large. Figure 6.10 depicts the RTT as a function of $n$, where (a) is the result of DyCW and (b) is the result of ARED using ECN. That is, DyCW improves downlink/uplink fairness at the cost of RTTs.

Figure 6.9 shows that AQM$^{smac}$ can achieve the similar fairness as DyCW. The RTTs and total goodput are similar to that of ARED. Buffer size has less impact on the ability of AQM$^{smac}$ and DyCW.

6.4.2 Simulation 2: Pure UDP flows

This section aims to show that the performance of DyCW is sensitive to packet size. We do simulations by varying $n$, buffer size and packet size. We only do simulations with AQM$^{smac}$ and DyCW. Figure 6.11 and Figure 6.12 show the results when packet size is 1500packets and 500packets respectively. In each figure, the left column is about DyCW; the right column is about AQM$^{smac}$. The results show that (i) the performance of DyCW is
sensitive to packet size; (ii) AQM_{smac} is not sensitive to packet size; however it sacrifices bandwidth utilization when n is not less than 15. The reason is as follows. In such network scenarios, differentiated ECN-marking is not effective for improving fairness since UDP flows are non-adaptive. Then AQM_{smac} shifts to adjust contention widow size of AP. As the number of mobile nodes is increased, the collision probability increases, resulting in the decreased total goodput.

6.4.3 Mixed TCP+UDP

We compare legacy DCF and AQM_{smac}. Each flow sends TCP and UDP traffic. The total sending rate of all UDP flows is 2Mbps, equally distributed among all the flows. Figure 6.13 shows the results. In these scenarios, AQM_{smac} depends on A^2RED to achieve the fairness.
6.4.4 Dynamic network

This section aims to investigate the performance of legacy DCF, DyCW and $\text{AQM}^{\text{smac}}$ in dynamic networks. $n=m=15$. Packet size is 1000 bytes. In the first 100s, all uplink flows are UDP and all downlink flows are TCP. In the 2nd 100s, all flows are TCP. In the 3rd 100s, all downlink are UDP and all uplink are TCP. In the 4th 100s, all flows are UDP. In the 5th 100s, each uplink flow and each downlink flow send both TCP and UDP traffic. Figure 6.14 depicts the results.

The results show that in this scenario $\text{AQM}^{\text{smac}}$ effectively improve the uplink/downlink fairness without sacrificing the link utilization. We also observe the insufficiency of $\text{AQM}^{\text{smac}}$ in the 1st 100 second in terms of improving fairness.
Figure 6.9  Average Goodput under AQM^{mac}

Figure 6.10  Average RTT versus \( n \)
6.5 Conclusions

This chapter first investigates whether using a modified queue-based AQM scheme with ECN-marking to manage IFQ queue can control congestion at the AP. The simulation results show that when there is no aggressive non-adaptive downlink flows, the modified queue-based AQM effectively keeps the IFQ queue length around the target value and hence the delays experienced by the flows are also low.
Subsequently, we investigate whether using a modified queue-based AQM scheme with ECN-marking can alleviate uplink/downlink unfairness. The simulation results show that using differentiated ECN-marking to notify of congestion can effectively prevent the greedy behavior of uplink TCP flows when the total sending rate of all UDP flows lies...
below the maximum available wireless capacity. In order to deal with the aggressive uplink UDP flows, we propose to combine the AQM scheme and smart MAC protocol. Extensive simulation results show that this approach outperforms other mechanisms in the literature over a wide range of network scenarios in terms of the fair distribution, small delay. Note that in some scenarios, the utilization is decreased due to collision.
Figure 6.14  Goodput variation versus time
CHAPTER 7
SUMMARY AND FUTURE WORK

In this chapter, we first summarize the work presented in this thesis, then give some possible directions for future work.

7.1 Summary of contributions and results

Packet-switched networks will continue being dominant in next-generation networks. The widespread deployment of the Internet results in the heterogeneous network environments and increasingly varying and unpredictable traffic conditions. Therefore, a mechanism without adaptability hardly meets the QoS requirement under such conditions. The first issue of concern in this thesis is congestion control, which is an essential problem in the network environment. We propose a stable queue-based adaptive AQM scheme. Stability analysis of the non-linear system is carried out.

Then we investigate the issue of providing QoS to users. We restrict our attention to investigating the existing QoS framework. We think that such investigation can benefit improving the performance of the existing framework and also benefit designing a new QoS framework. We use a control theoretic approach to analyze and compare various existing intelligent traffic conditioners in the AF-based DiffServ framework. Observing their disadvantage in terms of improving bandwidth assurance from the control theoretic point of view, we propose an adaptive method to adjust marking threshold. We also carry out the local stability analysis of the under-subscribed AF-based DiffServ network. This analysis reveals the relationship between network parameters and the marking threshold. It also suggests the importance of maintaining the stable queue length to guarantee bandwidth assurance. Then we validate our conclusions by experimentally investigating the effects of some existing AQM schemes on the performance of the AF service and the ability of intelligent schemes. The results show the importance of a stable system and the importance of transient response of AQM schemes.

Different from the wired networks, the congestion at the AP in the WLANs is caused by both uplink and downlink traffics. We revise the control-theoretic model developed in
the wired networks and then use it to analyze IFQ dynamics. A modified queue-based AQM scheme is proposed to handle the congestion at the AP. This AQM also effectively improve the uplink/downlink fairness. However the improvement is affected by the buffer size. Then we propose to assign different ECN-marking probabilities to downstream and upstream traffic. As long as the total sending rate of all non-adaptive flows is less than the maximum available wireless capacity, such differentiated ECN-marking can achieve uplink/downlink fairness. In order to further deal with the situation where there exit the aggressive non-adaptive uplink flows, we propose to combine varying the contention window with differentiated ECN-marking. Simulation results show that the proposed approach is very effective in meeting such objectives.

7.2 Future work

So far AQM_{mac} is implemented at the base station. In order to reduce the workload at the base station, it is better to move AQM_{mac} to gateway. However, we still require the base station to periodically convey the IFQ queue length. Does AQM_{mac} still maintain its ability in such situation? We only do simulations when AP is the only congestion point. Does AQM_{mac} still perform well when there is more than one congested point? In addition, how about the performance of AQM_{mac} in real WLANs? These deserve part of my short future work.

I also think of some long-term future works as follows:

- Resource Management and QoS in Wireless Networks. Certain application in industry networks has very stringent requirement for QoS, e.g. less than 1ms delay and less than 0.1ms delay jitter. Current schemes can hardly meet such requirement and I also would like to investigate what I have done in the wired networks to this issue in the future in the wireless field.

- Overlay Networks: My past research mainly focused on IP-layer algorithm design and the problems are addressed at the transport, the network, or even the MAC layers. With the popularity of overlay networks, in the future I will also look into the issues from the perspective of the application layer at the end systems.
References


138


