On efficient quality of service provisioning in the next generation internet

Siew, Chee Kheong

2007

Siew, C. K. (2007). On efficient quality of service provisioning in the next generation internet. Doctoral thesis, Nanyang Technological University, Singapore.

<https://hdl.handle.net/10356/39052>

<https://doi.org/10.32657/10356/39052>

On Efficient Quality of Service Provisioning in the Next Generation Internet

Siew Chee Kheong

School of Electrical and Electronic Engineering

A thesis submitted to the Nanyang Technological University
in fulfilment of the requirement for the degree of
Doctor of Philosophy

2007

Acknowledgements

I am very thankful for the assistance that I received during the journey that I have been through in this research, without which I am sure I would not be able to complete this work smoothly. First, I want to thank the Almighty God who has enabled me to do this intensive work during this part of my life. Without strength and wisdom that come from Him, I would not be able to come out of the wilderness and circling the mountain experience that I felt when I did my literature review. Secondly, I would like to thank my supervisor Prof Er Meng Hwa who is always encouraging me to complete this work as soon as possible. I would like to thank the following colleagues: Dr. Feng Gang, Dr. Tang Junhua and Dr. Huang Guangbin, who challenge my research ideas, Mr. Long Fei who assisted in plotting the graphs and others who assisted in one way or another.

I would like to thank my family, especially my wife Josephine who has been so supportive all these years in my life without which I would not be able to put so much attention in completing this thesis. I would like to my children Joanna, Jonathan and Joel for being such wonderful children so that I could wonder in my own world of intense research. I would like to thank my colleagues in ICIS who have been very supportive.

Finally yet importantly, I would like to thank NTU who provides me the opportunity to pursue a dream of a lifetime. This thesis would not be possible if I am not working in such an inspirational environment. My best wishes to university in her future endeavor.

Statement of Originality

I hereby certify that content of this thesis is the result of work done by me and has not been submitted for a higher degree to any other University or Institution.

15-2-07
Date

Siew Chee Kheong

Table of Contents

ACKNOWLEDGMENTS	ii
STATEMENT OF ORIGINALITY	iii
SUMMARY	ix
LIST OF FIGURES	xii
LIST OF TABLES	xiv
LIST OF ABBREVIATIONS	xv
LIST OF MATHEMATICAL SYMBOLS	xvii
1. INTRODUCTION -----	1
1.1. Motivation -----	1
1.1.1 Packet Switching versus Circuit Switching Networks-----	3
1.1.2 Two schools of Thoughts -----	4
1.1.3 Major Related Work -----	5
1.2. Objectives -----	6
1.3. Contributions -----	7
1.4. Organization of the Thesis -----	9
2. BACKGROUND -----	11
2.1. Introduction -----	11
2.2 Packet Scheduling -----	12
2.2.1 Fair Queuing -----	13
2.2.2 Round-robin Schedulers Priority Schedulers -----	18
2.2.3 Priority Schedulers -----	20
2.2.4 Flow and Packet Classification -----	21
2.2.5 Service Curves Theory -----	24
2.3 Traffic Shaping -----	26
2.4 Admission Control -----	27

2.5 Congestion Control	29
2.5.1 Congestion Problem	29
2.5.2. TCP Congestion Control based on End-to-End Mechanism	31
2.5.3. Router Assistant Congestion Control--- Active Queue Management (AQM)	33
2.5.4. Router Assistant Congestion Control—Congestion Control in Packet Scheduling Paradigm	35
2.6 QoS Architecture	36
2.6.1. Control Plane	36
2.6.2. Data Plane	41
2.7 Existing QoS Frameworks	43
2.7.1. Integrated Services (IntServ)	44
2.7.2. Differentiated Services (DiffServ)	44
2.8 Summary and Outstanding Issues	47
 3. A FLOW-STATE-DEPENDENT DYNAMIC PACKET SCHEDULING ALGORITHM FOR SERVICE CURVES ASSURANCE	 49
3.1 Introduction	50
3.2 Flow-based Dynamic Scheduling (FDPS) Algorithm	53
3.2.1 Conformant and –Non-conformant Traffic in FDPS	53
3.2.2 The FDPS Algorithm	54
3.2.3 Operation of FDPS and Analysis	58
3.2.4 Operation Example	63
3.3 Proof of FDPS Realization of Service Curves	66
3.4 Analysis and Properties of FDPS	70
3.4.1. Analysis of FDPS using Service Curve	70
3.4.2. Admission Control for FDPS	74
3.4.3 Admission Control Example	76
3.4.4 Difference between RCSP and FDPS	79

3.4.5 Comparison of Computational Complexities	80
3.5 Simulation Experiments	81
3.5.1 A simple Simulation topology	81
3.5.2 A More Complicated Simulation Scenario	88
3.6 Conclusion	91
4. CONGESTION CONTROL BASED ON FLOW-STATE-DEPENDENT DYNAMIC PRIORITY SCHEDULING	93
4.1 Introduction	94
4.2 Overview of the Proposed Source Aware Congestion Control (SACC)	97
4.3 Mechanism and Algorithm of Source Aware Congestion Control (SACC)	
Based on FDPS	100
4.4 Implementation and Evaluation of SACC Scheme	106
4.4.1 Simulation Settings	106
4.4.2 Experiment 1: Non-Regulated VBR Video and TCP Connections	108
4.4.3 Experiment 2: Regulated VBR video and TCP connections	111
4.4.4 Experiment 3: Non-Regulated VBR Video and TCP Connections with RED Gateways	113
4.5 The Behaviors of SACC in Best-Effort Traffic Environment	116
4.6 Conclusions	123
5. A NEW MULTISERVICE PROVISIONING MECHANISM WITH SERVICE CURVES ASSURANCE FOR PER-CLASS SCHEDULING DELAY GUARANTEES	125
5.1 Introduction	125
5.2 Multiservice Mechanism for Per-flow Delay Guarantee based on Class-based Traffic Treatment	131

5.2.1	Overview of the Proposed Multiservice Mechanism -----	131
5.2.2	Classification of Flows -----	133
5.2.3	Link Capacity Allocation -----	135
5.2.4	Packet Scheduling -----	136
5.2.5	The FDPS Algorithm -----	138
5.3	Analysis of the Scheduling Mechanism -----	140
5.3.1	Arrival Curve -----	140
5.3.2	The Service Curve -----	141
5.4	Admission Control -----	143
5.4.1.	An overview of Bandwidth Broker -----	143
5.4.2.	Bandwidth Reservation -----	144
5.4.3	Admission Control Analysis -----	145
5.4.4	Effects of Capacity Allocation -----	147
5.4.5	Admission Control Examples -----	150
5.5	Validation of the Proposed Multiservice Mechanism by Simulation -----	154
5.6	Conclusion -----	158
6.	A MULTISERVICE FRAMEWORK BASED ON THE FDPS ALOGIRTHM --	160
6.1	Introduction -----	160
6.2	Multiservices Framework -----	165
6.2.1	Service Specifications -----	166
6.2.2	Architecture -----	167
6.3	Summary -----	176
7.	CONCLUSION AND RECOMMENDATION FOR FUTURE WORK-----	179
7.1	Summary and Contribution -----	179
7.2	Recommendation for Future Work -----	182

Author’s Publications ----- 184

REFERENCES ----- 185

SUMMARY

The interest to provide quality of service (QoS) in the Internet has led to a plethora of research work that aims to provide QoS guarantee for many emerging applications in the Internet paradigm. As the Internet is based on the packet switching technology, the research work includes a wide range of topics, such as packet scheduling, admission control, flow and packet classifications, traffic policing, buffer management, etc. The accumulation and development of the research work in this area has led to two QoS architectures: Integrated Service (IntServ) and Differentiated Service (DiffServ). Integrated Service aims to provide per flow deterministic QoS guarantee but suffers from the issue of scalability. Differentiated service is proposed to address this issue by means of flow aggregation and service is provided by means of a per-aggregate per-hop behavior. However, only coarse QoS and service differentiation is provided in Differentiated Service.

There are two schools of thoughts concerning QoS provisioning. It appears that a group in the Internet community thinks that deterministic or quantitative QoS guarantees for the Internet is not necessary. They argue that most real-time applications could tolerate some losses without affecting user's perceived quality. It would be sufficient to provide stochastic guarantees in the Internet paradigm. A coarse QoS guarantee could be sufficient for all Internet users. We agree with part of this argument that a large group of Internet users need only coarse QoS guarantees. But, we argue that there is a distinct group of users who perform high value financial transactions today. This is the group who is utilizing expensive leased line service for their remote operations. A coarse QoS guarantee is not acceptable for this group of users. Due to the high monetary value in each transaction, these users would not accept probabilistic guarantees for the execution of their transactions. Unless deterministic guarantees could be provided in the Internet, it would be almost impossible to convince this group of users, who could pay a much higher subscription rate for deterministic QoS guarantees, to migrate to the Internet.

In this thesis, we address the issue of efficiently providing a range of per-flow QoS guarantees in the Internet paradigm. After a review of the relevant work on QoS guarantees, we propose a Flow-state-dependent Dynamic Priority Scheduling (FDPS) algorithm that provides Service Curves assurance. In this scheduling scheme, admitted QoS-based flows are protected yet allowing best effort traffic to co-exist in a multi-service network environment. The proposed FDPS scheme performs packet monitoring, marking, scheduling and discarding. With its fine granularity in packet marking, individual packets are forwarded and scheduled in a controlled and orderly manner. By means of mathematical analysis, we show that the FDPS algorithm realizes the arrival and service curves required in Service Curves and thus allows the theory of Service Curves to be applicable in a real network for per flow QoS provisioning. We also present an example for the admission control policing associated with FDPS. Compared with existing scheduling algorithms for QoS provisioning, FDPS achieves a tight end-to-end delay for QoS flows in a multi-service packet switching network. The complexity of this algorithm is independent of the number of flow, facilitating high scalability and possible implementation. Experimental results based on NS-2 validate the theoretical analysis and demonstrate the superiority of the proposed scheme.

Next we deeply study the behaviors and performance of FDPS algorithm for congestion control. We develop a new mechanism, namely Source Aware Congestion Control (SACC), in which congestion is handled at the packet scheduling layer. The SACC mechanism is based on FDPS, which performs traffic monitoring, marking and scheduling at routers. We find that that the FDPS algorithm enforces roughly a max-min allocation the bandwidth. TCP congestion collapse due to any TCP-unfriendly flow is prevented. We have shown that it can inter-work with existing active queue management schemes such as RED without modifications. Eliminating the need for setting the traffic dependent minimum and maximum threshold for Random Early Detection (RED) via the FDPS mechanism simplifies the congestion control mechanism.

For efficient utilization of bandwidth, we propose a new class-based multiservice mechanism with Service Curves assurance for per-flow deterministic delay guarantees. In

this mechanism, FDPS is employed to handle different flows within the same class to provide an end-to-end class-based scheduling delay bound guarantee. With the fine granularity in this mechanism, we are able to provide per-flow end-to-end delay bound independent of the number of intermediate nodes. In the above three proposals, we have provide mathematically analysis and conducted extensive simulations to verify the performance of a flow to verify the enhancements.

Based on the traffic control mechanisms as building blocks, we develop a new Multiservice (MultiServ) framework that addresses the issue of provisioning deterministic services with different scheduling delay bounds together with best effort service in a multiservice network. Recognizing that unless equivalent lease line QoS is available, total integration of all services onto the NGI would seem unlikely. The purposed framework has the following features: 1) It provides an equivalent leased line services together with best effort services; 2) It is simple and scalable; 3) It provides different QoS guarantees to different applications efficiently; 4) It permits gradual implementation; 5) It allows inter-domain QoS provisioning; and 5) It inter-works with existing congestion protocols and mechanisms such as TCP and RED without modifications.

Finally we conclude this thesis by highlighting our contributions in this PhD research project and pointing out some interesting potential issues for future research.

LIST OF FIGURES

- Figure 2.1 Curves of $A_i(0,t)$, $S(0,t)$, $Q_i(t)$ and $D_i(t)$
- Figure 2.2 An example of potential function in and RPS
- Figure 2.3 Deficit round-robin: at the start.
- Figure 2.4 Deficit round-robin: after serving flow #1.
- Figure 2.5 A connection involving three DS-domains
- Figure 3.1 Arrival and service curves
- Figure 3.2 Structure of a FDPS scheduler
- Figure 3.3 Schematic diagram of the Enhanced Token Bucket.
- Figure 3.4 Two sets of arrival and service curves
- Figure 3.5 A series of network elements of flow i
- Figure 3.6 The equivalent shaper envelope and service curve.
- Figure 3.7 Service curves for flows x and y .
- Figure 3.8 Simulation topology 1
- Figure 3.9 Packet delay in FDPS (Case A: two flows, $\sigma - \rho$ traffic)
- Figure 3.10 Packet delay in FDPS (three flows, On-off traffic)
- Figure 3.11 Simulation Topology 2
- Figure 3.12 Packet delay in FDPS (three flows, $\sigma - \rho$ traffic)
- Figure 3.13 Packet delay in FDPS (three flows, on-off traffic)
- Figure 4.1 Theoretical and practical traffic envelopes
- Figure 4.2 Simulation network topology
- Figure 4.3 Bottleneck queue length (VBR video + TCP flows)
- Figure 4.4 Packet loss ratio for drop tail, WRR and SACC scheme (VBR video+TCP)
- Figure 4.5 Packet delay statistics for video and TCP connections using drop tails, WRR and SACC schemes
- Figure 4.6 Packet loss ratio for drop tail and SACC scheme (regulated VBR video+TCP)
- Figure 4.7 Packet delay statistics for video and TCP connections using drop tails and SACC schemes

Figure 4.8 Comparison of bottleneck Queue Length with and without RED gateways

Figure 4.9 Comparison of packet loss ratio of video traffic with and without RED gateway

Figure 4.10 Comparison of packet delay statistics with and without RED gateway

Figure 4.11 Bandwidth shares of three TCP flows using FIFO with tail-drop router (Three TCP flows)

Figure 4.12 Bandwidth shares of three TCP flows using FDPS in the router (Three TCP flows)

Figure 4.13 Bandwidth shares of three TCP flows using FIFO with tail-drop router (1 CBR UDP flow + two TCP flows)

Figure 4.14 Bandwidth shares of three TCP flows using FDPS in the router (1 CBR UDP flow + two TCP flows)

Figure 4.15 Bandwidth shares of three UDP flows (CBR) using FIFO-Tail-drop in the router

Figure 4.16 Bandwidth shares of three UDP flows (CBR) using FDPS in the router

Figure 5.1 The Multiservice Scheduling mechanism

Figure 5.2 Network topology used in the simulation experiments

Figure 5.3 Comparison of CCDF for scheduling delays of class 1, class 2 and beset effort traffic (Experiment 1)

Figure 5.4 Comparison of CCDF for scheduling delays of class 1, class 2 and beset effort traffic (Experiment 2: an additional class 2 video flow is admitted)

Figure 5.5 Packet loss when one class 2 flow includes non-conformant traffic

Figure 6.1 The multiservice framework architecture

LIST OF TABLES

Table 3.1 Experiment parameters of the token bucket of the FDPS scheduler

Table 3.2 Maximum end-to-end delay for two cases ($\sigma - \rho$ TRAFFIC)

Table 3.3.a Experiment parameters of the token bucket of the FDPS scheduler (case A)

Table 3.3.b Experiment parameters of the token bucket of the FDPS scheduler (case B)

Table 3.4 Maximum end-to-end delay for two cases (ON-OFF TRAFFIC)

Table 3.5 Traffic parameters for the on-off traffic

Table 3.6 Maximum end-to-end delay for two cases

Table 4.1 Experiment parameters of the token bucket of the SACC scheduler

Table 5.1 Flow Parameters

Table 5.2 Experiment parameters of the token bucket of the FDPS scheduler

Table 6.1 Comparison of IntServ, DiffServ and MultiServ Framework

LIST OF ABBREVIATIONS

AF	Assured Forwarding
AIMD	Additive Increase and Multiplicative Decrease
AVQ	Adaptive Virtual Queue
AQM	Active Queue Management
CBFQ	Credit-based Fair Queuing
DiffServ	Differentiated Services
DSCP	Differentiated Services Code Point
DRR	Deficit Round-Robin
EDF	Earliest Deadline First
EF	Expedite Forwarding
ERR	Elastic Round Robin
ETB	Enhanced Token-Bucket
FCFS	First-Come-First-Serve
FDPS	Flow-state-dependent Dynamic Priority Scheduling
GPS	Generalized Processor Sharing
IntServ	Integrated Services
IP	Internet Protocol
ISP	Internet Service Provider
NGI	Next Generation Internet
OSPF	Open Shortest Path First
PHB	Per-Hop Forwarding Behavior
PGPS	Packetized Generalized Processor Sharing
PRPS	Packetized Rate-Proportional Server
QoS	Quality of Service
RCSP	Rate Control Static Priority
RED	Random Early Detection
REM	Random Exponential Marking
RPS	Rate-Proportional Server
RSPEC	Receiver service request specification
RSVP	Resource Reservation Protocol

RTT	Round-Trip-Time
SLA	Service Level Agreement
TCP	Transmission Control Protocol
TB	Token Bucket
TOS	Type of Service
TSPEC	Traffic Specification
VFT	Virtual Finishing Time
VoIP	Voice-over-Internet-protocol
WF2Q	Worst case Fair weighted Fair queuing
WFQ	Weighted Fair Queueing

LIST OF MATHEMATICAL SYMBOLS

$A(t)$	arrival curve
$A_i(t)$	traffic arrival of flow i at time t
$A_i(0, t)$	traffic envelope of flow i in interval $[0, t]$
$B(t)$	the set of all backlogged flows at time t
$I(t)$	the set of active sessions at time t
L_{max}	maximum packet size
$V(a_i^k)$	virtual arrival time of packet k of flow i
$W_i(t_1, t_2)$	service provided to session i
ϕ_i	weight assigned to flow i
r_i	rate of flow i
σ	burstiness of the flow
ρ	long term rate of the flow
t	time interval
$\bar{A}_i(t)$	conformant traffic of flow i
$\tilde{A}_i(t)$	non-conformant traffic of flow i
σ_i	depth of flow i 's leaky-bucket
ρ_i	reserved rate of flow i
$\bar{S}_i$	shaper envelope of flow i
$S_i(t)$	minimum service curve of flow i
$\hat{A}_i(t) = \sigma_i + \rho_i t$	arrival curve of flow i with leaky bucket traffic specification
d_j	delay elements for link j
$B_i(t)$	backlog of flow i at time t
$F * G(n)$	convolution of F and G using min-plus algebra
$LB(\sigma_i, \rho_i)$	leaky bucket of flow i with parameters σ_i and ρ_i

d_i^s	maximum scheduling delay of flows i
δ_{d_i}	delay element i
C_j	capacity of link j

CHAPTER 1

INTRODUCTION

1.1 The Motivation

With the ubiquitous access of the Internet, numerous emerging applications have sprung up in many parts of the world today, employing the Internet to carry multiple types of traffic. The Internet is inherently designed as best effort datagram networks due to uncoordinated resource sharing. Traditionally, in the Internet all packets are treated the same without any discrimination or explicit delivery guarantees. This is known as the *best effort service model*; all the network promises is to exert its best effort to deliver the packets injected into it without committing to any quantitative performance (Quality of Service, QoS) bounds.

However, recent developments in the diversity of application requirements make it obvious that a diverse range of services delivered to the users becomes imperative. The last decade has witnessed very rapid growth of the Internet accompanied by an evolution of new applications, ranging from complex applications including Videoconferencing, IP telephony, interactive multimedia, distributed gaming etc, to simple data services. The former class of applications often requires Quality of Service (QoS) guarantee, which cannot be provided by the current IP network architecture based on the Best-effort service model. This new requirement calls for an architecture that can support QoS while preserving the scalability and simplicity of the current Internet.

Previous efforts to provision QoS in the Internet have resulted in the Integrated Service (IntServ) [BR94] and the Differentiated Services (DiffServ) [BL98] models. The purpose of IntServ is to provide per-flow absolute QoS guarantee. However, due to its poor scalability and a lagging demand for per-flow absolute QoS, the interest in Internet QoS eventually is shifted to the DiffServ architecture, which was proposed as a scalable

solution based on the same paradigm as the Internet. This architecture employs simple packet scheduling and buffer management at the core routers based on tags that are set at the edge routers. The edge routers are allowed to perform traffic management on a per-flow basis while the core routers are not. In this architecture, the edge routers are responsible for the end-to-end service assurance. Although some traffic management and packet marking mechanisms have been proposed for the edge routers, whether DiffServ is able to deliver the end-to-end service differentiation or QoS guarantee across applications is not clear.

The need to provide QoS in the Internet paradigm has led to a plethora of research work in QoS provisioning in the literature. One of the key interests is to design the Next Generation Internet (NGI) [ROB96] to integrate all services. To this end, it would have to provide deterministic guarantee, stochastic guarantee and best effort services in the same network. Before further discussion on what the NGI can provide, we provide a brief review of the key issues affecting the Internet carrying real-time traffic. In the current Internet, the path of a flow consists of concatenations of many links that provide the connectivity between its source and destination. The capacity available to a flow depends on the conditions of the instantaneous traffic at each of these links. With the first-come-first-serve scheduling policy, the resulting capacity of path is a stochastic process that depends mainly on the behavior of the traffic at the bottleneck link in the path. For example, a user who subscribes to a 100 Mbps link from a local Internet Service Provider (ISP) could be receiving a service of 30 Kbps due to the bottleneck in the path of its connection. It would be more economical for this customer to subscribe to a cheaper 1 Mbps local link without any degradation of performance. In other words, over subscription at the local access network does not imply high end-to-end path capacity. This is a necessary but not sufficient condition for good performance. The complex interactions of cross flows at every link along the path of a flow would determine its instantaneous path capacity. In the next section, we provide a comparison of packet switching and circuit switching networks.

1.1.1 Packet Switching versus Circuit switching Networks

In circuit switching, a channel is dedicated to each flow for the period of connection. Clearly, this is not an efficient way to allocate bandwidth for a variable rate flow that may idle for part of the time, a common behavior of most connections. Although the dedicated bandwidth in circuit switching has many desired properties such as guaranteed rate, isolation from other traffic and constant end-to-end delay, it suffers from low bandwidth efficiency. This is the key issue that leads to the development of packet switching. In contrast to circuit-switching, packet switching as applied in the Internet provides a virtual channel to each flow. The physical channel could be shared by a large number of users, each using only a small fraction of the capacity over a short duration. The multiplexing gain in packet switching has made it the desirable technology since 1970s. An active flow could utilize this channel at any time based on first-come-first-serve (FCFS) scheduling. It is recognized that FCFS scheduling does not provide any isolation among flows sharing a link. It allows the behavior of one flow to affect the characteristic of other flows after the interaction at a shared link. As a result, the burstiness of a flow increases with the shared links traversed. We will discuss this process involving flow interactions in more details.

The dedicated channel in circuit switching provides an ideal condition that isolates one flow from another. If this flow misbehaves and starts transmitting aggressively above traffic specification, it would affect its own performance with no effect on other flows. On the other hand, the misbehavior of a flow has immediate consequences on the burstiness of other flows sharing the same link in the Internet. One may suggest that policing traffic at ingress nodes for compliance would solve this problem. We can show that this is not the result. Suppose we regulate every flow at the input by a shaper to transmit only in-profile traffic into the Internet, a flow would still acquire additional burstiness at every link along its path. Such accumulation of burstiness would affect flows sharing downstream links. It is shown in a recent research [BE02] that an arbitrary delay can be exceeded for a flow with a few hops if the network has a large number of hops. Without proper isolation of traffic for each real-time flow, allocating the same

bandwidth as those in a circuit switching network is not sufficient to guarantee the same performance to these flows due to accumulation of burstiness as a result of the FCFS scheduling.

A natural question to ask is this: What is QoS of a flow? We address this question in details in relation to the quest of using the Internet for real-time services. In general, service rate, delay, jitter and loss are the four key parameters defined to quantify the QoS for a flow. Service rate refers to the rate guaranteed to the flow. This often represents the reserved rate of the flow for the period of connection. Delay refers to the maximum delay tolerable by a flow. The actual delay of a flow is the sum of propagation delays, processing delays, transmission delays and queuing delays that its packets encounter. If the maximum delay is exceeded, a packet reaching its destination is deemed unusable and counted part of the traffic loss. Jitter [VE91] is a measure for delay variation due mainly to the stochastic variations in flow bandwidth due to the sharing process. A flow would experience very low jitter in a dedicated channel. Jitter is normally resolved by means of play-out buffering at the receiver. It would facilitate continuous play-out of information without buffer starvation that affects the perceived quality. Loss is a probabilistic measure of the traffic loss experienced by the flow. As mentioned earlier, this loss is mainly due to delay exceeding the maximum delay allowable by a flow under the assumption of large buffers. The actual parameters considered in QoS provisioning depends on whether a flow is loss tolerant or not. This led to the extensive studies on deterministic and stochastic QoS guarantees in the Internet paradigm. The former refers to quantitative service guarantee while the latter refers to probabilistic service guarantees respectively. In quantitative QoS guarantees, hard bounds, at which violation is not allowed, are specified. Whereas, stochastic guarantee uses soft bounds which define the limits of possible violation based on some probabilistic measure.

1.1.2 Two Schools of Thoughts

There are basically two schools of thoughts concerning QoS provisioning. It appears that a group in the Internet community thinks that deterministic or quantitative QoS guarantees for the Internet is not necessary. They argue that most real-time applications

could tolerate some losses without affecting user's perceived quality. It would be sufficient to provide stochastic guarantees in the Internet paradigm. A coarse QoS guarantee could be sufficient for all Internet users. We agree with part of this argument that a large group of Internet users need only coarse QoS guarantees. But, we argue that there is a distinct group of users who perform high value financial transactions today. This is the group who is utilizing expensive leased line service for their remote operations. A coarse QoS guarantee is not acceptable for this group of users. Due to the high monetary value in each transaction, these users would not accept probabilistic guarantees for the execution of their transactions. Unless deterministic guarantees could be provided in the Internet, it would be almost impossible to convince this group of users, who could pay a much higher subscription rate for deterministic QoS guarantees, to migrate to the Internet. In other words, deterministic QoS guarantee is needed in the NGI for it to become the network that integrates all services. We will review whether recent works can meet this requirement.

1.1.3 Major Related Work

Accumulation of recent research work on QoS provisioning leads to two well-known architectures: 1) Integrated Services (IntServ) [BR94] and 2) Differentiated Services (DiffServ) [BL98]. The IntServ architecture is proposed to provide deterministic per-flow QoS guarantees. It addresses the notion of QoS in the Internet paradigm by means of resource reservation along the path of a flow, shaping traffic at the ingress node and packet scheduling at each node. It provides strict service guarantee in which in-profile packets of a flow are protected and delivered within their delay bound. A shaper dimensioned according to the traffic parameters of the flow is normally added at the ingress node to regulate the traffic of each of the applications that require deterministic QoS guarantees. Admission control at the ingress node would ensure that sufficient path capacity is available before a flow is admitted. A resource reservation protocol is normally used to reserve resources such as bandwidth and buffer at nodes along the path of the requested flow. Due to the large number of flow states in this architecture and its computational complexity, physical implementation is not feasible for a large number of flows.

In response to this limitation, DiffServ [BL98] is proposed to address the issues of computational complexity and scalability in IntServ. It addresses these issues via aggregation of flows into a limited number of classes and provides per aggregate per hop behavior service. This architecture moves the complexity from the core to the edge of a network. As an edge node handles a lower number of flows, more intelligence can be implemented at edge nodes without affecting the performance of the network. Three types of service are provided in this architecture: 1) Premium Service; 2) Assured Service; and 3) Best Effort Service. DiffServ provides service differentiation and a coarse QoS guarantee by means of the premium service. As mention earlier, it has been proven [BE02] that the delay of a flow could exceed an arbitrary large delay bound after passing a few links under certain conditions if the network has a large number of nodes. It is clear that DiffServ would provide a rough QoS guarantee depending on the network configuration and traffic characteristics. For the Internet to become the network that integrates all services, further enhancement is needed to provide deterministic QoS guarantees that provide a range of delay guarantees efficiently.

1.2 Objectives

As mentioned at the beginning of this chapter, the current Internet provides only best effort service. Together with the rapid growth in the number of applications running over the Internet, many commercial applications have tried to provide cheap audio and video services over the Internet. Unless the service provided by the NGI could satisfy users' QoS requirements provided by other networks today, it would be difficult to convince all users in other networks to migrate to the Internet. We identify that different types of traffic would require different types of QoS differentiation. It would be desirable to provide a range of deterministic QoS guarantee to cater to different types of traffic ranging from equivalent leased line service to best effort service. In contrast to over provisioning to provide QoS guarantees, an efficient and easy to implement solution is needed in the Internet. The proposed solution must be highly scalable and has low computation complexity for physical implementation. To the best of our knowledge, there is little work on an implementable solution that provides deterministic QoS

guarantees. In particular, a mechanism that provides a range of quantitative QoS guarantees efficiently in the NGI is needed. This mechanism must cater to deterministic guarantee that allows ISP to charge different subscription rates for different guarantees with best effort service at the bottom of the range of services. It should work with current congestion control schemes, in particular, the Transmission Control Protocol (TCP) congestion control schemes. Since a connection in the NGI could pass through a number of autonomous domains before it reaches its destination, there is a need of a framework that facilitates inter-domain quantitative QoS guarantees.

1.3 Contributions

The contributions of this thesis are summarized as follows:

- 1) The first contribution is the proposal of a novel scheduling algorithm called Flow-state-dependent Dynamic Priority Scheduling (FDPS) that provides deterministic per-flow QoS guarantees for the Internet paradigm. Unlike most scheduling algorithms in the literature, memory of unconsumed guaranteed service of a flow affects the way packets are scheduled. In the same manner, out-of-profile traffic are identified by the algorithm and treated as best effort traffic. This is a new algorithm that provides Service Curves assurance. Service Curves is a well-known theory that provides a theoretical approach for deterministic QoS guarantee. Besides providing a mathematical analysis for the FDPS algorithm, we have conducted extensive simulations using NS-2 to verify the mathematical results. We have shown that this algorithm could provide deterministic per-flow QoS guarantee in the packet switching paradigm.
- 2) The second contribution of this work is the study of the effect of FDPS algorithm in congestion control. The FDPS algorithm allocates negative priority to non-conformant (out-of-profile) traffic. These best effort packets are then scheduled according to their priority. Congestion control is normally handled at the transport layer. Many research works were performed to study the behavior of TCP mechanisms. TCP utilizes an additive increase multiplicative decrease rate

adjustment to facilitate fair sharing of bottleneck links. It is shown that the issue of fairness is affected by the Round-Trip-Time (RTT) of each flow and flows with a shorter RTT get more bandwidth at a bottleneck link than long RTT flows. The implementation of the FDPS algorithm for packet scheduling resolves this throughput dependency on RTT, an interesting result for the TCP protocol. We have shown that the FDPS algorithm enforces roughly a max-min allocation of the bandwidth. TCP congestion collapse of any TCP-unfriendly flow is prevented. We have shown that it can inter-work with existing active queue management schemes such as Random Early Detection (RED) without modifications. In addition, our NS-2 simulations show that similar performance for TCP over FDPS is achieved, independent of RED. Eliminating the need for setting the traffic dependent minimum and maximum thresholds for RED via the FDPS mechanism simplifies the congestion control mechanism.

- 3) Class-based traffic treatment frameworks such as DiffServ have been proposed to resolve the poor scalability problem in the flow-based approach for service differentiation and QoS provisioning. Although the performance in these frameworks is differentiated on a class basis, how to provide delay guarantee for individual flows in the same class has not been well understood and the issue has not been adequately investigated. In this thesis, we present a new multiservice mechanism with Service Curves assurance for per-flow deterministic delay guarantees. In this mechanism, the novel Flow-state-dependent Packet Scheduling (FDPS) is employed to handle different flows within the same class to provide an end-to-end class-based scheduling delay bound guarantee. With the fine granularity in this mechanism, we are able to provide a per-flow end-to-end delay bound independent of the number of intermediate nodes. We mathematically analyze the performance of a flow using this class-based FDPS to demonstrate the enhancements. We also use simulation experiments based on NS-2 to validate the proposed mechanism.
- 4) In the fourth contribution, we propose a new Multiservice (MultiServ) framework that addresses the issue of provisioning deterministic services with different

scheduling delay bounds together with best effort service in a multiservice network. Recognizing that IntServ provides deterministic QoS but has unresolved scalability and complexity issues, and DiffServ provides coarse QoS guarantees, requirements of small group important users who require leased line service are still unaddressed. Unless equivalent lease line service is available, total integration of all services onto the NGI would seem unlikely. The research work in this project faces the following challenges: 1) To design a framework that provides an equivalent leased line services together with best effort services; 2) It should be simple and scalable; 3) It should be able to provide differentiated services to various applications with specific guarantees efficiently; 4) It permits gradual implementation; 5) It allows inter-domain QoS provisioning; and 6) It inter-works with existing congestion protocols and mechanisms such as TCP and RED without modifications. We have proposed an efficient practical solution for such provisioning using class based FDPS algorithm. The works have cumulated into a Multiservice QoS framework that provides end-to-end deterministic per-flow QoS guarantee in a multi-domain environment. We have contributed towards the understanding of how deterministic per-flow QoS guarantee could be provided in a class-based mechanism. With this enhancement, we could use the NGI to carry traffic with QoS requirements ranging from equivalent leased line service to best effort service. In this mechanism, best effort traffic is not subject to admission control but the FDPS algorithm facilitates the rough sharing of bandwidth according to a max-min scheme and works with current congestion control mechanisms such as RED without modifications.

1.4 Organization of the Thesis

The rest of the thesis is organized as follows. Chapter 2 provides a review of various algorithms and common architectures related to QoS provisioning. Chapter 3 presents our new proposal - the FDPS algorithm and its features. We demonstrate how per-flow QoS could be provided in a packet switching domain by using FDPS. Chapter 4 presents how the features of FDPS scheduling interoperate with the transport layer mechanisms, including the TCP congestion control algorithm and Active Queue Management (AQM)

schemes. We show that the FDPS algorithm enforces rough sharing of the bottleneck link according to a max-min fair share scheme. In Chapter 5, we show quantitative per-flow QoS guarantee could be provided in a class-based FDSP mechanism that provides Service Curves assurance. Chapter 6 presents a Multiservice QoS framework by using FDPS as a main building block. This is a generic framework for end-to-end per-flow QoS guarantees for multi-domain connections. Finally Chapter 7 presents the conclusion of the thesis and recommendation for future work.

CHAPTER 2

BACKGROUND

2.1 Introduction

There is a plethora of research works on QoS provisioning and best effort traffic available in the literature today. However, the solution of providing per-flow deterministic QoS guarantees in the Internet paradigm seems to be elusive to the Internet community. Many accepted the notion of coarse QoS guarantees as the solution for the Internet paradigm. We have provided a comprehensive argument in Chapter 1 that it is highly desirable to integrate all services in the Next Generation Internet (NGI) [ROB96]. With the multiplexing gain via integrating all services onto a packet switching NGI, Internet Service Providers (ISP) and users could enjoy higher profit and lower cost of services. Hence, it is highly desirable to provide both deterministic and non-deterministic QoS guarantees in the NGI. We will show that unless deterministic QoS guarantee is available in the NGI, it would be almost impossible to attract those existing and new users, who would be willing to pay for leased line quality, to migrate to the NGI. Though a theoretical solution based on Service Curves for deterministic QoS guarantees [CR91, CR91+, CR96, CH94, SAR99, SAR95, AG99, BOU01, BOU96] is available today, there is little work on physical implementation with these theoretical bounds. The search for a practical solution for quantitative guarantee in the Internet paradigm is still an open research issue. Due to the large number of works in these areas, we would only present the major research works carried out with QoS provisioning as the main objective. The general approach to provide QoS guarantees includes the following processes: 1) packet scheduling [PAR93, PAR94, ZH93, ZH95, ZHL90, GOY97, GOY97+, ST96, BE96, BEN01, SI02, GO94, HA91, STO00, STO98, SU97] 2) traffic shaping [GE97, GE96, GE96+], and 3) admission control [KN99, KN97, KN98, KN97+, LIE96, MY03, SA97,

CE01, EU03]. Most of the work in the current literature assumes that a separate routing process pre-selects the path of a flow where we apply QoS provisioning. With this selected path, we examine how these components interact with one another and provide QoS guarantee to a flow. Next, we discuss in more details the above three processes and related works in the following sections. We would then refer to these processes in subsequent chapters without providing further details.

The rest of this chapter is structured as follows. In Section 2.2, we describe some well-known scheduling algorithms and their operations. In Section 2.2, we present the concepts of traffic shaping and relate the concepts to Service Curves theory. In Section 2.3, we present admission control needed for QoS guarantees and interesting findings in the literature. In Section 2.4, we present a review on congestion control in the best effort paradigm. In Section 2.5, we present a review on QoS architectures. The discussion includes both the control plane and data plan functions. Some of the interesting protocols and mechanisms are also presented. In Section 2.6, we present existing QoS architectures: Integrated Service and Differentiated Service. In Section 2.7, we provide a summary of the chapter and discuss the limitations of the current architectures that lead to a gap between the requirements of the users and the capability of the Internet. As a result, the current Internet could not realize the multiplexing gain due to its inability to integrate all services onto the Internet.

2.2 Packet Scheduling

There is a plethora of research work in packet scheduling [PAR93, PAR94, ZH93, ZH95, ZHL90, GO97, GO97+, ST96, BE96, BEN01, SI02, GO94, HA91, STO00, STO98, SUR97] based on packet switching in the Internet. The current Internet provides a simple best-effort service. In this service, the network does not provide any guarantee: the packet delivery process could corrupt, delay or lose packets. End processes at each host would have to recover such packets, usually performed by the transport layer. The current Internet provides the best effort service based on a first-come-first-serve scheduling. It is simple to implement but could not provide the protection needed for service guarantees unless we apply very large bandwidth over provisioning. As discussed

in Chapter 1, this efficient and simple scheduling scheme induces more burstiness on a flow at every shared link in its path. As a result, many researchers aim to find a scheduling algorithm with properties that match its users' expectations. To this end, one of the main issues in early research work in the Internet is how to schedule packets to sharing a link fairly among competing flows, commonly known as fair queuing. In fair queuing, a backlogged flow receives its fair share of the link if the quantum of service it receives is proportional to the ratio of its reserve rate to the sum of the reserved rate of all backlogged flows at the link, the idea of weighted fair queuing (WFQ) [PAR93, PAR94, ST96]. The idea develops further into a guaranteed rate model [GOY97+] that guarantees a specific rate to a flow, independent of the behavior of other flows. For this model to work, admission control must ensure that there is sufficient capacity in the path of a new flow request before admitting a new flow. Further research work is conducted to provide rate and delay guarantees. This leads to interest on providing rate and delay isolation and network proposals that move away from the best effort paradigm. The key idea behind scheduling is to serve packets in an order that meets some predefined objectives. The difficult issues are whether this is possible and how this could be done remain the objectives of many recent works. In the remainder of this section, we present well-known algorithms related to QoS provisioning.

2.2.1 Fair Queuing

The main objective of this set of scheduling algorithms is fair sharing of a link bandwidth among all backlogged flows. In the ideal situation, each flow should receive a fair share of the bandwidth at any moment in time. Perhaps the Generalized Processor Sharing (GPS) algorithm in [PAR93] lays the theoretical basis for weighted fair queuing in the Internet. GPS is a fluid based work-conserving algorithm that assumes it is possible to service all backlogged flows simultaneously. It assumes that all flows could be serviced by an infinitesimal amount and the service provided to any two flows i and j satisfies the following inequality:

$$\frac{S_i(\tau, t)}{S_j(\tau, t)} \geq \frac{\phi_i}{\phi_j} \quad (2.1)$$

where ϕ_i and ϕ_j are weights assigned to flow i and j respectively, subjected to a link capacity limitation as defined in the following inequality [PAR93]:

$$\sum_{i \in I(t)} r_i \leq C \quad (2.2)$$

where $I(t)$ is the set of active sessions at time t , r_i is the rate of flow i , and C is the link capacity.

Specifically, the rate guaranteed to each session i is

$$g_i = \frac{\phi_i}{\sum_{j \in B(t)} \phi_j} r \quad (2.3)$$

where $B(t)$ is the set of all backlogged flows at time t and r is the rate of the server. Usually, only part of the active flows is backlogged at a time, that is, $B(t) \leq I(t)$.

As fluid-based scheduling is a theoretical concept, Parekh proposed the packetized GPS algorithm known as PGPS [PAR94]. The idea in PGPS is to service packets in the ascending order of their virtual finishing time according to the GPS system. The virtual finishing time F_i^k for packet k of flow i is defined as follows:

$$F_i^k = \max\{V(a_i^k), F_i^{k-1}\} + \frac{L_i^k}{\phi_i} \quad (2.4)$$

where $V(a_i^k)$ is the virtual arrival time of packet k .

Parekh also introduced a leaky bucket $LB(\sigma, \rho)$, where σ represents the burstiness and ρ the long-term rate to regulate the burstiness of an individual flow. As a result, the envelope of the bucket $LB(\sigma, \rho)$ becomes the upper bound of the traffic of the flow. In addition, he pioneered the idea of service curves to provide an interpretation of the number of backlogged packets and the packet delay at a node.

Figure 2.1 Curves of $A_i(0,t)$, $S_i(0,t)$, $Q_i(t)$ and $D_i(t)$

Figure 2.1 [PAR93] depicts the traffic envelope $A_i(0,t)$ of flow i at time t , and the service curve $S_i(0,t)$ offered to flow i at time t . Due to the mismatch of these two curves, the amount of backlogged packets and delay experienced by a packet arriving at time τ are given by $Q_i(\tau)$ and $D_i(\tau)$. We would discuss concept of arrival and service curves in more detail in subsequent subsection 2.14.

The research community identifies that both GPS and PGPS as theoretical algorithms due to their high computational complexity in tracking the number of backlogged flows and the computation of the service rate of each flow using Equation (2.3). To compute the virtual system time in Equation (2.4) needed for the calculation of virtual finishing time of each packet, a simulator needs to track the system virtual time. Parekh also extended the idea of PGPS to a network of cascaded PGPS servers [PAR94] based on non-cut through switching. He derived the delay bound of the PGPS rate proportional sharing network as

$$D_i^{*,PGPS} \leq \frac{\sigma_i + 2(K-1)L_i}{g_i} + \sum_{m=1}^K \frac{L_{max}}{r^m} \quad (2.5)$$

where K is the number of servers along the path of flow i , σ_i is the depth of the leaky bucket, L_{max} is the maximum packet size at server m and r^m is the rate of server m .

In [BE96], Bennet *et. al.* revisited the question of fairness in PGPS. They showed that PGPS and WFQ (GPS) are different. Contrary to the idea that delay bound provided by WFQ is within one packet transmission time of GPS, large discrepancies between the service offered by WFQ and GPS systems exist. He proposed an algorithm called worst-case fair weighted fair queuing (WF^2Q) to overcome this unfairness. In WF^2Q , a packet eligibility time, which denotes the time that this packet should have started transmission in a GPS system, is introduced. By scheduling the packet according to the earliest virtual finishing time computed by Equation (2.4) that has ought to have started service in the GPS system (current time not less than eligibility time), the service provided by WF^2Q and GPS differs no more than one maximum packet size.

Other work which approximates the system virtual time [GO94, GOY97] aims to reduce the computational complexity of PGPS. These works utilizes the finishing-time or the start-time of the current packet to approximate the system virtual time. After computing the packet virtual finishing time (VFT) using Equation (2.4), the packet at the head of each flow are sorted in the increasing order of their VFTs and the packet with the lowest VFT is scheduled. As a result, the order of complexity is reduced to $O(\log N)$ where N is the number of backlogged flows.

Stiliadis [ST96, ST98] proposed a fluid-based fair queuing algorithm known as Rate-Proportional server (RPS). In RPS, he introduced the concept of a connection potential $P_i(t)$ of a session i as

$$P_i(t_2) - P_i(t_1) = \frac{W_i(t_1, t_2)}{\rho_i} \quad (2.6)$$

where $W_i(t_1, t_2)$ is the service provided to session i . A RPS scheduler serves only the set of sessions with the minimum potential values. For multiple sessions with the same potential, their service rates are proportional to their reserved rates (see Figure 2.2). This

scheme aims to equalize the potential of each backlogged session, thus allocating service proportional to their reserved rates.

Figure 2.2 An example of potential function in and RPS

In [ST98], the system potential function has two important properties:

$$P(t_2) - P(t_1) \geq (t_2 - t_1), \quad (2.7)$$

for a system busy interval $(t_1, t_2]$. In Figure 2.2,

$$P(t) \leq \min_{j \in B(t)} \{P_j(t)\} \quad (2.8)$$

where $B(t)$ is the set of all backlogged connections.

Stiliadis extended the fluid-based RPS to a packetized RPS known as PRPS in [ST96]. In PRPS, the system potential and a connection potential are inputs for computing a timestamp for each arriving packet. The PRPS algorithm schedule packets in the increasing order of their timestamps. In [ST98], tracking the global state of the system precisely requires simulating the corresponding fluid based RPS in parallel with the packet-based system. It is not difficult to observe the similarities between the RPS model and the GPS model as both are proportional rate servers in which the service offered to each backlogged connection is proportional to the ratio of its reserved rates to the sum of

the rates of backlogged flows. In [ST96], an approximation for the virtual system potential is proposed, a problem similar to GPS and PGPS.

Although the above approximations have reduced the computational complexity of the PGPS and PRPS algorithm, it is not obvious whether the computational complexity $O(\log N)$, a feature due to sorting of N packets, is feasible for millions of flows serviced at core routers. The search for a practical implementation for fair queuing among backlogged connections leads to the development of round-robin schedulers in the next section.

2.2.2 Round-robin Schedulers

Round-robin schedulers [SHR96, YAM02, KA02, KAT91, HA91, CHA03, LEN04, GUO04] form a group of simple schedulers with low computational complexity, sometimes as low as $O(1)$. This has attracted much interest in the research community to apply this algorithm for packet scheduling. Many of these research works [CHA03, LEN04, GUO04] are very recent. The Deficit Round-Robin (DRR) [SHR96] scheme utilizes a per-flow deficit counter to implement fair queuing. The deficit counter of each backlogged flow is incremented by a service quantum in each round robin cycle. This limits the amount of service that a flow would receive in each cycle.

If the size of the packet at the head of a flow is smaller than the deficit counter, the algorithm schedules this packet and decrements the deficit counter decremented by the amount of service provided. Otherwise, it processes the next queue in the same manner (see Figures 2.3 and 2.4). In order that each flow could transmit one packet in each round of assignment, the service quantum must be equal to the maximum packet size Q_{max} of the link. On the downside, increasing the quantum Q_i leads to increasing the scheduling delay of small packets [SHR96]. By reducing the quantum Q_i , more queues may have to wait for the accumulation of the sufficient credit in the deficit counter before their packets get transmitted.

Figure 2.3 Deficit round-robin: at the start.

Figure 2.4 Deficit round-robin: after serving flow #1.

Other versions of round-robin schedulers include Elastic Round Robin (ERR) [KA02] and Weighed Round Robin [KAT91]. ERR is a modified version of the DRR in which the deficit counter (DC) is allowed to become negative temporarily and the quantum Q allocated to each flow is proportional to the bandwidth reserved. It has similar properties as DRR except that in every round, every flow whose DC is positive will be served, an improvement over DRR. The idea is similar to credit based fair queuing (CBFQ) [BEN01] with a fixed packet size for scheduling. In this case, both of them have a complexity of $O(1)$. Most of the recent works [CHA03, LEN04, GUO04] on round robin are to improve the latency or smoothness of the algorithm.

It is worth noting that these research works on fair queuing do not consider users' delay requirements. Two applications with the same bandwidth requirement may have quite different delay requirements. In general, a round-robin algorithm provides rate guarantees without delay guarantee. The downside of fair queuing is the high dependency between rate and queuing delay. To reduce queuing delay for a flow, one could apply rate over provisioning, an inefficient method of service guarantee. From this early experience in fair queuing, the research community continues the search for a simple and implementable algorithm for rate and delay isolation. To this end, attention is focus on priority scheduling discussed next.

2.2.3 Priority Schedulers

Priority scheduling [BER98, II01, ZH93, ZH95] is a simple and scalable algorithm for a reasonable number of priorities. A priority scheduler [ZH93] normally maintains one queue per priority for output buffering. This differs from the one queue per flow in the case of fair queuing discussed in the previous section. The scheduler forwards traffic of the same priority to the same queue to wait for service. For example, if an implementation has P queues for priorities 1 to P , where 1 and P represent the lowest and highest priorities respectively. When the output link is available, the priority scheduler always schedules the backlogged queue with the highest priority. This

algorithm has low computational complexity and simple to implement. Due to the manner service is provided, lower priority traffic is not isolated from higher priority traffic. Central to this algorithm is how to assign the appropriate priority to each packet. Fundamentally, there are two types of priority scheduling: static priority [BER98, II01, ZH93, ZH95] and dynamic priority [SI02, HU93]. In static priority, a classification process, which we would discuss in more details later, assigns a priority to a flow during connection establishment. All packets belonging to this flow would carry the same priority for the duration of the connection. In general, high priority flows encounter very low end-to-end delays at expense of low priority flows. In dynamic priority scheduling, the priority of different packet of a flow could be different depending on the dynamics of the control variables in the algorithm. These could include link capacity, flow rate, queue length and link occupancy. In other words, the priorities of packets belonging to a flow could change during the duration of the connection. Once priority is assigned to a packet, it is forwarded to an output queue according to its priority for priority scheduling. In the remainder of this section, we would discuss the flow and packet classifications.

2.2.4 Flow and Packet Classification

There are few published works on the mapping of flow or packet classification to quantitative QoS guarantees. The reason could possibly be due to the difficulty in mapping classification parameters such as source and destination address and port numbers, and protocols to quantitative QoS guarantees in terms of rate, delay, jitter and loss. More research on classification algorithms is available. But, our primary interest is in the mapping and not classification algorithm. From these limited works, we have presented some of the interesting observation below.

Flow classification is normally implemented in the control plane. Flow classification is one of the functions in admission control executed during connection establishment. Once a flow is classified, the same priority would be assigned to all packets of the flow during the duration of the connection. This priority does not change as the packets travel from the ingress node to the egress node. A simple classification policy is to classify

flows according to protocols [LIN97]. In this policy, hop-by-hop forwarding is provided to UDP flows while ATM virtual channels support TCP flows. It assumes that connection oriented TCP flows would last longer than connectionless UDP flows. Another alternative policy is to classify flows according to application. A different flow classification policy utilizes a static X/Y/T algorithm [HAO99], where X represents the number of packets that arrives in an interval Y and T is the time elapses without any packet arrival would trigger the termination of a flow. A flow is considered a long-lived flow when more than X packets arrive in interval Y. In the adaptive algorithm [HAO99], a reasonable interval is assigned to Y and X is adjusted dynamically according to the characteristics of traffic, processor load and cache. This is a simple identification scheme that identifies a flow as a long-lived flow or otherwise. Other adaptive schemes [WA01, ZHE99] aim to improve the adaptation by maximizing the number of active label-switched long-lived flows or adjusting the value of X in X/Y/T algorithm to maximize the virtual circuit space and have a larger number of flows switched by the ATM hardware. Flow classification is also applied in Differentiated Services (DiffServ) in which all flows would be classified into one of its three class and all flows of the same class are aggregated together. The DiffServ mechanism marks the differentiated services code point (DSCP) field of a packet for the purpose of service differentiation. The priority scheduler accesses the DSCP field in a packet to determine the priority for scheduling. Common flow-based classifications parameters include: 1) Type of service; 2) Source and destination addresses and ports; 3) Importance of information; and 4) Protocols. It is worth noting that there is a very large combination of these parameters available for classifications. A simple example of flow classification involves the following order of priority: voce communications, video communications followed by web traffic. To map a combination of parameters to a quantitative QoS for flow classification is still an open issue for research. Although it is not obvious how this could be done, a recent work [SC03] shows how a flow could be assigned to a particular priority (class) that provides the appropriate queuing delay bound.

In general, the data plane of the network performs real-time packet classification. Based on the same parameters, packets with the same characteristics are aggregated together

into flows and provided the same treatment (e.g. service differentiation and security treatment). In a broad sense, the VFT in fair queuing [PGPS] could be considered a packet classifier to which scheduling service is accorded. In this case, there is an infinite number of priorities that a packet could be assigned.

A third method of classification involves both the control and data plane. In this method, the control plane performs a flow-based classification followed by packet-based classification at the data plane. This method adds a new degree of freedom not available to the earlier two methods. For example, Huang [HU93] proposes a dynamic priority scheme for ATM network using multiple priority levels. If the time limit L_i is exceeded, the relative priority of a cell is raised above cells of other classes and is selected for transmission in the next time slot. In another example, Yasukawa [YAS04] apply the idea of dynamic class assignment for streaming flows depending on traffic characteristics of non-streaming flows. With non-streaming flows being classified in one of its four classes, a new streaming flow would be dynamically classified into a class with sufficient capacity, allowing for better balancing of load among the classes. Streaming flows are also assigned higher drop precedence than non-streaming flows. However, class-to-service mapping is still not obvious. From these works, we concluded that priority scheduling appears to be a promising method for implementation due its simplicity and low computational complexity.

Further research in rate and delay guarantees leads to the development of deterministic QoS service provisioning. As a result, the Service Curve [CR91, CR91+, CR95, CH94, SAR99, SAR95, AG99, BOU01, BOU96] theory is proposed to address the issue on traffic protection and how service should be provided to a flow for deterministic QoS guarantees. These issues are discussed in the next section.

2.2.5 Service Curves Theory

In the last two sections, we discuss the possible implementations of fair queuing and priority service algorithms. In this section, we examine the theoretical aspects of how service could be provided given the characteristics of a flow. Parekh first applied the idea of service curve in fair queuing [PAR93, PAR95] as discussed in section 2.2.1. In [CR95], Cruz applied this theory to provide service guarantees in virtual circuit switched networks. Referring to Figure 2.1, the upper bound of delay and backlog of flow i are given by $\max_{\tau \geq 0} D_i(\tau)$ and $\max_{\tau \geq 0} Q_i(\tau)$, respectively. The point of intersection of $A_i(0, t)$ and $S_i(0, t)$ is the time when all backlogged packets are served. The arrival traffic envelope $A_i(0, t)$ defines the upper bound of the traffic arrival while the service envelope $S_i(0, t)$ defines the minimum service provided to flow i . If $S_i(0, t) = A_i(0, t), \forall t$, then every packet is serviced when it arrives and no queuing delay is incurred. When $S_i(0, t) < A_i(0, t), \forall t$, queuing delay is incurred. The upper bound of this delay is given by

$$d[t] = \min \{ \Delta, \Delta \geq 0 \text{ and } A_i(0, t) \leq S_i(0, t + \Delta) \} \quad (2.9)$$

Sariowan [SAR99] introduced a framework for service curves in which scheduling policies such as Virtual Clock and the Earliest Deadline First (EDF) could be classified as special cases. A service curve defines the minimum service that the network could provide that best matches the QoS requirement by a user. Using the definition of service curve in [SAR99], a network element guarantees a service curve $S(\cdot)$ for a flow i if for any time t , there exist $s \leq t$ such that $r^{out}(t) - r^{in}(s) \geq S(t - s)$, where $S(\cdot)$ is a non decreasing function and $S(0) = 0$. If the traffic envelope of a flow i is b -smoothed, $A_i(t_1, t_2) \leq b(t_1, t_2)$ and the server guarantees service curve given by $S(\cdot)$, the upper bound of the backlog $B_i(t)$ is given by [SAR99, Theorem II]:

$$B_i(t) \leq \max_{s \geq 0} \{ b(s) - S(s) \} \quad (2.10)$$

It is necessary to reserved sufficient buffer to prevent any packet loss due to queuing. For the case of a flow with an upper bound delay of d^{max} and b -smooth, it is obvious

that d^{max} is guaranteed if the server guarantees a service curve $S(x) = b(x - d^{max})$. If the backlog is bounded by B^{max} , both requirements will be met if the server guarantees the service curve [SAR99]

$$\hat{S}(\tau) = \max \{b(\tau - d^{max}), [b(\tau) - B^{max}]^+\} \quad (2.11)$$

where $x^+ = \max \{x, 0\}$.

In [SAR99], it claimed that SCED scheduling policy provides a means for a server to allocate service curves to different flows to meet their requirements. It guarantees the service curve $\lfloor \hat{S}_i(\cdot) \rfloor$ to session i for $i = 1, 2, \dots, M$ if the following condition is satisfied

$$\sum_{i=1}^M (b_i * \hat{S}_i)(t) \leq C(t) \quad (2.12)$$

where $F * G(n)$ is the convolution of F and G in min-plus algebra [BOU01]. Service curves present a flexible way to meet the QoS of all connections subject to Equations (2.11) and (2.12).

In [CR98], the idea of a minimum service curve $S(t)$ and maximum service (arrival) curve $\bar{S}(t)$ is introduced. As maximum service curve is the upper bound of the service provided to any flow, the output stream R^{out} conforms to the arrival curve b^{out} given by [CR98],

$$b^{out} = (b^{in} * S) \circ S \quad (2.13)$$

where b^{in} is the arrival curve of R^{in} and $X \circ Y$ means function X deconvolved with function Y .

Based on these concepts of minimum and maximum service curves, Cruz [CR98] introduced damper to all flows that "slow down" any packet that have arrived too early by means of an eligibility time. By assigning an initial eligibility time $e_k^{initial}$ and a terminal eligibility time $e_k^{terminal}$, where $e_k^{initial} \leq e_k^{terminal}$, packet k departs immediately if it arrives at or after its terminal eligibility time. Otherwise, the damper ensures that the

departure time of packet k satisfies $e_k^{initial} \leq t_k \leq e_k^{terminal}$. This idea is similar to [SUR97] where it slowed down early packets. The purpose of the damper is to give better service to the best-effort traffic, subject to not compromising the required service guarantees for other traffic. Thus, this system provides service to “early” guaranteed traffic only if there is no best-effort traffic. The idea of redirecting service to other packets and delaying the service to early high priority packets presents a method for satisfying the QoS for more flow, worth considering in future works. However, including a damper could introduce a processing bottleneck when the damper may release a large number of packets in the same slot. Further investigation would be helpful to quantify how the interactions of these packets would affect the packet delay of each flow. In particular, redirecting service from packets that are early to those that are late could provide a new approach for efficient utilization of the Internet for QoS provisioning. In conclusion, this theory [CR91, CR91+, CR95, CH94, SAR99, SAR95, AG99, BOU01, BOU96] has provided the foundation for the investigation of deterministic QoS guarantees and evaluation of other algorithms.

However, there is little work on its physical implementation of Service Curves possibly due to the difficulties in tracking the arrival and service curves of a flow. A similar study on the arrival curve takes the form of traffic shaping before traffic enters a node. Traffic shaping modifies the characteristics of the flow to conform to the shaper envelope very useful for admission control as discussed in section 2.3. In the next section, we present the effect of shaping and reshaping of the traffic of a flow in a network.

2.3 Traffic Shaping

We observed a parallel development of research in traffic shaping [GE97, GE96, GE96+] in the context of QoS provisioning. The basic approach of shaping a flow at the entrance to a packet switching network does not preclude the accumulation of burstiness due to the interactions between flows at shared links. In other words, a flow transmitting in-profile traffic into a network would interact with other flows at shared links along its path, leading to a rise in its burstiness with number of links traversed. Under the

assumption that admission control, which would be discussed in the next section, ensures that there is sufficient capacity for admitted flows, Georgiadis [GE97, GE96] reshapes the traffic at every hop. They proposed a per-flow per-hop shaping for every flow to limit the burstiness of each flow at every node. This reshaping at every hop limits the burstiness of a flow to its characteristics at its ingress node. He shows that reshaping based on the same envelope does not incur additional delay [GE97, GE96, CR98, AG99] besides the delay due to the first shaper. This is sometimes referred to as “reshaping is free [PE97]”. This work is closely related to the theory of Service Curves as reshaping at every hop is equivalent to defining an identical arrival curve at every hop. There is no additional delay incurred if the envelope of the arrival curve at each hop of a flow is identical. The delay of a flow could be guaranteed by this interesting scheme. An intuitive explanation to this per-flow per hop reshaping is as follows. Admission control ensures that there is sufficient capacity at every link for all flows admitted under this scheme. By reshaping a flow at the input of every node, a flow that acquires additional burstiness at a previous node would be smoothed by the shaper to its original characteristics at input of the previous node. As a flow’s token bucket is always ready to transmit in-profile traffic, reshaping does not incur additional delay. However, reshaping reverses the effect of packets clumping due to the interactions at a shared link. Sufficient path capacity provisioned by admission control would ensure that a packet is serviced at the same rate at every node along its path thus providing an equivalent “constant rate pipe”. The natural question that follows is this: “What is the minimum capacity that admission control must provide ? ” In the next section, we would attempt to answer this question from the current literature.

2.4 Admission Control

While a packet scheduler decides, which backlogged packet to schedule according to one or more parameters, admission control [FI98, KN99, KN97, KN98, MY03, SA97, CE01, SAH00+] decides whether a new flow could be admitted. The goal of admission control is to ensure that there are sufficient resources in the network to support a new connection without violating the QoS of existing flows before it is admitted. Otherwise, it

would reject the new flow. We would discuss the interdependent relationship between packet scheduling and admission control in the Internet paradigm. Admission control ensures that the necessary resources are reserved before it admits a new flow. The resource requirements depend on the requirements of scheduling algorithm and the QoS required by a flow. Consider an example of a network using a WFQ algorithm: the reserved rate depends on the delay of the flow; a high reserved rate is needed for a low queuing delay. This is a case where queuing delay is inversely proportional to reserved rate. As a result, the scheduling algorithm determines the reserved rate needed to provide the QoS of a flow.

In general, the criteria of admission control could be divided into two types of guarantees: 1) stochastic guarantee; and 2) deterministic guarantee. It is worth noting that best-effort traffic is not subject to any admission control due to the non-guaranteed nature of the delivery. In stochastic guarantee [KN99, KN97, KN98], the idea is to admit a flow with guarantee based on a predefined low probability of traffic discard. In other words, a probabilistic guarantee is provided to these flows admitted in this manner. The statistical properties of flows (e.g. rate-variance method [KN99]) are often utilized in these research works to admit the maximum number of flows subjects to the probabilistic guarantee provided. Measurement-based admission control [SA97, CE01] is also studied by some researchers. In this method, the algorithm would monitor the actual traffic condition all the time. Based on this information the admission control mechanism could admit more flows due to actual capacity available, applicable to stochastic QoS guaranteed only. For the case of deterministic guarantee [FI98, BR94], full protection would be provided to admit flows based on the worst-case scenario. In this case, absolute quantitative guarantees for rate and delay are provided to a flow admitted and none of its packets would be lost due to traffic related reasons. One of the interesting implementations is the admission control for DiffServ architecture via bandwidth broker [TE99, DU04, QB]. We would provide a separate discussion for bandwidth broker in section 2.5. For the current research, our focus is deterministic QoS guarantees while stochastic QoS guarantees are beyond the scope of the current work. Next, we would present in more details the development of admission control for deterministic guarantees.

In [FI98, FI98+, AG99, CR98], the idea of service curves is applied in admission control. In this control plane function, the requirement of a new flow is encapsulated by its service curve derived from its arrival curve. As discussed in section 2.14 on Service Curves, the arrival curve of a flow is dimensioned by the traffic parameters of the flow. It represents the envelope of the in-profile traffic of the flow. The idea in [FI98] is to subtract the required service curve from the remainder of the service capacity of a link to ascertain whether there is sufficient capacity for a new flow. If part of the new curve is negative, it would not be feasible to admit the new flow and yet provide the deterministic guarantees to all guaranteed flows. Service Curves remains a theoretical method for service provisioning with no simple method for implementation. One of the main difficulties in Service Curves is the tracking of the arrival and service curves in real-time. As the NGI would carry best effort traffic, a review on how congestion control for best effort traffic in the current Internet is given in the next section.

2.5 Congestion Control

In this section, we first revisit the network congestion problem and examine the end-to-end congestion control mechanisms based on TCP. Next, we will discuss the router-assistant congestion control mechanisms - AQM and their infeasibility in dealing with non-responsive or non-TCP compatible traffic.

2.5.1 Congestion Problem

Congestion can be defined from a network perspective or a user perspective [PAN01]. From the network perspective, congestion is defined as the state of sustained network overload where the demand for network resources is close to or exceeds network capacity including bandwidth, buffer space and processing power in the routers.

The Internet has suffered from the problem of congestion which is inherent in best effort datagram networks due to uncoordinated resource sharing. Traditionally, in the Internet all packets are treated the same without any discrimination or explicit delivery

guarantees. This is known as the best effort service model; the network promises to exert its best effort to deliver the packets injected into it without committing to any quantitative performance (QoS) bounds. There is no admission control mechanism and the user can inject packets into the network without obtaining permission in a prior. Worst still is that there is no coordinated resource sharing mechanism in the best-effort service model. Therefore the perceived performance is determined not only by the network itself, but also from other users' offered load, resulting in a complete lack of isolation and protection. Though the traditional applications and protocols were flexible, adaptive, and robust enough to operate under a wide range of network conditions without requiring any particularly well-defined service, the problem of congestion has become more and more serious, along with the introduction of more applications to the Internet.

Congestion is the state of sustained network overload where the demand for network resources is close to or exceeds capacity. Network resources, namely link bandwidth and buffer space in the routers, are both finite and in many cases still expensive. Congestion occurs when the situation sustains that packets from multiple flows arrive at the router simultaneously and need to be forwarded on the same output link. Obviously, not all of them can be forwarded simultaneously; a service order is required. A buffer space must be provided as temporary storage for the packets waiting for transmission. In other words, busy traffic from multiple sources can create a demand for network resources higher than what the network can handle at a specific link. The buffer space in the routers offers a first level of protection against an increase in traffic arrival rate. However, if the situation persists, the buffer space is exhausted and the router has to start dropping packets. Packet dropping is a signal of network congestion. From the user's perspective, congestion is a notion related both to user's satisfaction and to network load. A network is said to be congested from the perspective of a specific user, if the satisfaction of the user decreases due to a modification of the performance (bandwidth, delay, jitter, etc.) of the network connection [PAN01]. Congestion in the Internet can cause high packet loss rates, increased delays, and even breakdown of the whole system by causing congestion collapse [LE02]. This is a state where any increase in the offered load leads to a decrease in the useful work done by the network.

2.5.2. TCP Congestion Control Based on End-to-End Mechanism

The Internet protocol architecture is based on a connectionless end-to-end packet service using the IP protocol. In order to deal with congestion the Internet used end-to-end window-based flow control in its Transmission Control Protocol (TCP) [SHE98, JA88], primarily for controlling demand on the receiver's bottleneck resources. TCP as the main transport protocol used in the Internet employs congestion avoidance and congestion control mechanisms to deal with the congestion problem. In congestion avoidance [JA88, CHI89], the idea is to monitor the state of the network and generate feedback to all sources sharing the bottleneck link to increase/decrease their rate according to the change of network states [JA88].

The TCP congestion control mechanisms have served the Internet quite well and formed the basis for its survival and success. The TCP window management algorithm uses the well proven principles of Additive Increase and Multiplicative Decrease (AIMD) [CHI89]. TCP Tahoe and Reno [SHE98, JA90] deal with the recovery for a congested node. In TCP Tahoe [SHE98], a lost packet triggers the reduction of the current window threshold to half of its current window size and reenters slow-start by resetting the window size to 1. To improve the utilization of the link capacity, a lost packet in TCP Reno [JA90] triggers the source to freeze the window size in the fast retransmit/fast recovery phase and reduces the window threshold to half of its frozen current window size at the end of this phase and enters congestion avoidance immediately. TCP Vegas [BR94] is an enhancement to TCP Reno. One of the main enhancements is the correction of oscillatory behavior in TCP Reno by keeping the number of packets buffered in the network between 1 and 3. The idea is to make better use of available network resources to improve network performance.

TCP family protocols are based on an end-to-end mechanism without the support from the intermediate routers. Specifically, the traditional technique for managing the queue at an intermediate router is the simple "drop-tail". A router accepts packets for its queue until the maximum length is reached, then drops subsequent incoming packets until the

queue decreases as a packet from the queue has been transmitted. This queue management scheme has two obvious drawbacks. First, a single connection or a few flows may monopolize the queue space, preventing other connections from getting room in the queue in some situations. Second, the tail drop discipline allows queues to maintain a full (or, almost full) status for long periods of time, since tail drop signals congestion (via a packet drop) only when the queue has become full. It is important to reduce the steady-state queue size and thus the average end-to-end packet delay, and this is perhaps queue management's most important goal.

For tail drop, two alternative queue disciplines that can be applied when the queue becomes full are "drop front on full" or "random drop on full". Under the latter discipline, a router drops a randomly selected packet from the queue when the queue is full and a new packet arrives. Under the former discipline, the router drops the packet at the front of the queue when the queue is full and a new packet arrives. Both of these disciplines solve the lock-out problem [IETF98], but neither solves the full-queues problem.

The performance of the Additive Increase Multiplicative Decrease (AIMD) algorithm [CHI89] for TCP congestion control has been well studied and understood in the literature. Some research work addresses the problem of TCP congestion control using model-based rate control [PAD99] and equation based congestion control [FL00]. Other research work [MO00, YA03] has been concentrated on fair end-to-end congestion control or the dynamic transient behavior of the congestion control/avoidance processes. These approaches to congestion resolution discussed so far focus on the TCP layer based on window or rate control to provide congestion avoidance or recovery. It is also proven that a flow becomes more bursty when it traverses more network nodes if the FIFO discipline is applied [BE02, ST96]. Such a characteristic would add to the likely-hood of traffic congestion. It is not clear whether congestion control at the TCP layer (transport layer) is the best solution as it relies on the lower layer to provide end-to-end connectivity.

2.5.3. Router Assistant Congestion Control - Active Queue Management

In the current Internet, dropped packets serve as a critical mechanism of congestion notification to end nodes. In order to cope with the increasing congestion in Internet due to an exponential increase in network traffic, the IETF has been considering the deployment router assistant congestion control mechanism, including Explicit Congestion Notification (ECN) [IETF01] and active queue management techniques (AQM) such as RED[FL93], AVQ [KU01] and BLUE [FE02], which can be used with a various scheduling algorithms to provide significant Internet performance improvement.

One form of congestion notification that has been discussed since the advent of RED is Explicit Congestion Notification (ECN) [IETF01]. ECN has been proposed to allow routers to help in congestion control by notifying users when it detects an onset of congestion. A router on detecting incipient congestion in a link sets a bit in the packet header that notifies the user that a link on its route is experiencing congestion. The user then reacts to the notification as if a packet has been lost. In this way, the link avoids dropping the packet (thereby enhancing goodput) and still manages to convey congestion information to the user. To provide ECN marks or drop packets to provide fairness and control queue lengths, the routers have to select packets in a wise manner that conveys information about the current state of the network to the users. Algorithms which the routers employ to convey such information are the Active Queue Management (AQM) schemes. An AQM scheme could mark or drop packets depending on the policy at the router.

The basic idea of AQM is that a router drops packets before a queue becomes full, so that end nodes can respond to congestion before a buffer overflow. By dropping packets before buffers overflow, active queue management allows routers to control when and how many packets to drop. An AQM mechanism can provide the following advantages for responsive flows: (1) reduce number of packets dropped in routers; and (2) reduce the delays seen by flows by keeping the average queue size small. In the past years, many AQM mechanisms have been proposed [FL93, FE02, PAN01].

Random Early Detection (RED) [FL93] is an active queue management algorithm for routers that can provide performance enhancement for the Internet. In contrast to traditional queue management algorithms, which drop packets only when the buffer is full, the RED algorithm drops arriving packets probabilistically. It uses the average queue length for congestion control to address the problem discussed above. As a result, if the queue has recently been relatively full, indicating persistent congestion, newly arriving packets are more likely to be dropped. Researches and implementations of RED illustrate that it can effectively control the average queue size while still accommodating bursts of packets without loss. The synchronized multiple TCP backoff processes are broken up by random early dropping of packets.

The adaptive virtual queue (AVQ) scheme [KU01] maintains a virtual queue whose capacity (called virtual capacity) is less than the actual capacity of the link. When a packet arrives in the real queue, the virtual queue is also updated to reflect a new arrival. Packets in the real queue are marked/dropped when the virtual buffer overflows. The virtual capacity at each link is then modified such that total flow entering each link achieves a desired utilization of the link. A feature of the AVQ scheme that is appealing is in the absence of feedback delays. It has shown that the AVQ algorithm performs better than a number of other well-known AQM schemes.

The Random Exponential Marking (REM) scheme [ATH01] attempts to achieve both high utilization and negligible loss and delay by decoupling congestion measure from performance measure. BLUE [FE02] uses packet loss and link-idle events rather than the queue length to control congestion. BLUE increases the packet drop probability in response to a buffer overflow (i.e., a packet drop) and decreases the packet drop probability when the link becomes idle.

In order to control bursty Internet traffic effectively and efficiently, most existing AQM proposals recommend using the (average) queue length as a congestion indicator. However, there remain several critical problems, such as mismatch between macroscopic

and microscopic behavior of queue length, insensitivity to the input traffic load and pattern variation and parameters-setting problem.

2.5.4. Router Assistant Congestion Control—Congestion Control in Packet Scheduling Paradigm

The legacy Internet architecture assumes that all flow-related states should be kept at the hosts; therefore, the congestion control mechanisms hitherto have been mainly implemented at the end hosts. Upon detection of congestion the sources should inject their packets into the network more slowly. In order for a host to be able to detect congestion, the routers should provide the information that the network is currently overloaded; this mechanism is called feedback-based congestion control. The feedback from the network and the response from the source are the foundations of Internet congestion control and are very important because they facilitate decentralized resource allocation. However, with decisions made at the end hosts and treatment of the network as a black box that simply drops packets, there is clearly a limit on control over the allocation of network resources. This also limits the range of services the network is capable of offering. Routers, on the other hand, know exactly how congested links are and can therefore perform more intelligent resource management. Thus, the introduction of router mechanisms for congestion control enables the network to manage its own resources more efficiently, which is a reasonable choice.

A previously proposed way to use network (router) support for congestion control is to change the scheduling discipline inside the routers. Fair Queuing scheduling [BE96] is well known for its flow isolation property. However, the research community does not agree on the utility of this scheduling discipline for congestion control, even if, its flow isolation property is appreciated, nor on the way to use this scheduling discipline [LE02]. The reasons include its high implementation complexity and the uncertainty on how WFQ can be used to handle best-effort and QoS traffic concurrently existing in the network.

Hitherto, we have provided the review on scheduling and admission control and congestion control. As mention in chapter 1, many of these works have cumulated into the proposal of two existing QoS architectures: Integrated Service and Differentiated Service. In the next section, we present in details on the components of a QoS architecture.

2.6 QoS Architecture

In this section, we revisit the goal of QoS provisioning, define the key parameters and present the components and their functions in this architecture. The goal of this architecture is to provide QoS guarantees to its flows. QoS guarantee can be divided into two types: 1) Deterministic guarantee and 2) Stochastic guarantee. For deterministic guarantee, an absolute quantitative guarantee is provided to a guaranteed flow. This provisioning is based on the worst-case scenario design. Whereas, in stochastic guarantee, a probabilistic guarantee is provide to a guaranteed flow. In this case, there is small probability of QoS violation for a guaranteed flow. By keeping this probability reasonably small, most real-time services are able to tolerate such packet loss (e.g. voice or video transmission) due to their inherent redundancy in their coding. It is recognized that providing a range of required QoS efficiently is a challenging task [ST96].

In general, service rate, delay, jitter and loss are the four key parameters considered in QoS guarantees. For deterministic QoS guarantee service, we consider only the first two due to no loss experienced by the flow in this service. Jitter, a measure of delay variation, is often resolved by appropriate buffering for each flow at the receiver. The fourth parameter applies to stochastic guarantees in which efficient utilization is achieved via allowing a small probability of traffic loss.

2.6.1. Control Plane

The architecture normally consists of a control plane and a data plane. The control plane handles the requirements in connection establishment and termination. The major functions in the control plane include: 1) Admission control; 2) Routing that selects the

end-to-end path; 3) Resources reservation; 4) Setting traffic policing, buffering and discarding policy at each node; and 5) Signaling for the establishment and termination of a connection. With details on admission control discussed in Section 2.4, we present routing and signaling functions next.

A. Routing and Signaling

These two functions normally come hand-in-hand in a connection setup. While the routing function selects the path according to a particular criterion (e.g. OSPF) implemented in the routing algorithm, signaling protocol [LIU00, MO92] handles the handshakes and distribution of state information that update the management tables at each node on the path of a new flow. Some of the common criteria for path selection include shortest path, widest path and fixed routing. The choice for implementation is service provider dependent. For example, a routing algorithm could select a particular path based on the special requirement of a flow. An indirect path with more hops instead of a single hop path could be selected due to the propagation delay requirement. The choice between a satellite hop and submarine optical fiber connection could be such a case. The state of the network or its abstraction in terms of various path cost must be available to ingress node for the selection of the most appropriate path under the current condition. The signaling protocol [LIU00] in the network must update the state of ingress node for efficient route selection. However, these state information changes rapidly due to the transient conditions in a large network. The issue of state maintenance to provide up-to-date information, especially to the ingress nodes, is a challenge in a large network such as the Internet. To this end, the invention of the Bandwidth Broker [QB] provides a possible solution to this problem by means of centralized state maintenance in an autonomous network domain. In contrast, Integrated Service utilizes a distributed state implementation, sometimes known as a stateful network. Once the route is selected, admission control activates a resources reservation protocol (such as RSVP [IETF97]) to reserve the resources required by the packet scheduling algorithm along the selected path. Upon successful reservation of needed resources, admission control accepts the new connection. We present a brief overview of both RSVP protocol and Bandwidth Broker in the following.

B. RSVP Protocol

The RSVP [IETF97] protocol is designed to reserved bandwidth along the paths of a multicast session. A unicast session could be considered as a subset of the multicast session. When a new multicast request arrives, admission control activates the RSVP protocol to place reservation for bandwidth at all links in the selected paths. Reservation signaling declares the traffic specification in terms of burstiness and long-term rate of the source to its receivers. This information is carried by a traffic specification (TSPEC) along the selected paths to its destinations. In response to the TSPEC information, each receiver sends its desired service (RSPEC) up the multicast tree to the source. The TSPEC is specified in terms of a token bucket $TB(\sigma, \rho)$ plus a peak rate (p), a minimum packet size (m) and a maximum packet size (M). The depth σ of the token bucket defines the maximum burstiness of the flow and the peak rate (p) limits the rate of its transmission. The RSPEC information is similar to the TSPEC information with two additional parameters: the desired bandwidth R and delay guarantee S . Although two services, guaranteed service and controlled load service, are specified in IntServ [BR94], only the formal is discussed here. RSPEC that converges at a link would be merged as it travels up the multicast tree. Each network element must confirm that it has sufficient capacity before the multicast flow is admitted. In this process, the states of the flows are updated at network elements of the multicast tree. These soft states are maintained throughout the period of the flow. It is worth noting that the basic RSVP deals with per flow states that could raise the issue of scalability when a large number of flows is involved.

C. Bandwidth Broker

The goal of the Bandwidth Broker [QB] is to develop a simple inter-domain QoS signaling for the purpose of the end-to-end QoS guarantee for a flow than spans across multiple autonomous network domains. Each domain may have different QoS mechanisms in a heterogeneous network environment. Although the BB architecture is designed for heterogeneous network domains, the design applies particularly well to the multiple DiffServ domains. In particular, the BB complements the DiffServ architecture

in administering the admission control function. With one BB per DiffServ domain, it could operate as its gatekeeper in admission control and negotiates with BB of an adjacent domain when a flow spans more than one DS domain. These functions are very useful for QoS provisioning in a DiffServ domain due to its stateless core nodes. The stateless core nodes enable scalability at a core node but admission control needs link and flow states for its decision. To this end, a domain specific BB would maintain the link and flow states (databases) in the domain, keeping core nodes simple. We would provide an illustration on the inter-working of BB for admission of a flow that spans multiple DiffServ domains as follows.

Figure 2.5 A connection involving three DS-domains

Figure 2.5 shows the path of a flow request from host S to host D that spans three DS domains. This signaling process could be extended to any number of DS domains. Host S resides on DS domain DS1 and the host D in DS3 with DS2 as the intermediate DS domain. Assume that host S requests a premium service connection through the three domains. To set up the connection, host S sends an RSVP PATH message to its bandwidth broker DS1-BB. Depending on the resources available and bandwidth policy, this BB would decide whether it could accept this request. In general, whenever a request is denied, an error message is sent back to the requester and this message is then propagated to the originating host and the signaling ends. When the request is accepted by DS1-BB, it sends an RSVP PATH message to DS2-BB. In accepting the request, DS2-BB would send an RSVP PATH message to DS3-BB. Upon accepting the request, DS3-BB would set classification and policing rules on router IR-3 and ER-3 and send an

RSVP RESV Message to DS2-BB. On receiving the RESV message, the DS2-BB would set classification and policing rules on IR-2 and ER-2 and send an RSVP RESV message to DS1-BB. On receiving the RSVP message, DS1-BB would set the classification and policing rules in IR-1 and ER-1 and send an RSVP message to host S. When S receives that RSVP message, it enters data transmission phase.

Note that there are significant differences in the BB signaling process when compared to the IntServ/RSVP process: 1) A source requests for resources instead of the receivers; 2) Any BB may reject the request when it receives the PATH message. Only receivers could reject a request in IntServ/RSVP. The BB architecture allows a BB to reject a request according to the bandwidth policy independent of the receiver and thus avoid tying up of valuable resources. A BB could aggregate multiple requests and make a single request to the next BB, enabling request scalability [QB]. Each domain behaves as a single virtual node, represented by a BB while core nodes in each DS domain are not involved in the signaling process. State information (databases) stored in BB, ingress and egress nodes in each DS domain are soft state information subjected to a time out. This state information must be refreshed regularly by the latest state information.

Though the BB architecture is still evolving, it has provided a viable inter-domain signaling platform for the DiffServ architecture with simple core nodes and stateful edge nodes. By maintaining the flow state database in the BB, the latest state information is available at the BB for efficient admission control that facilitates high scalability.

D. Flow Classification

The control plane during connection establishment normally performs flow classification. It utilizes the services provided of the scheduling algorithm to classify a flow into an appropriate class. Based on this classification, a per-class service would be provided to this flow, that is, all flows in the same class are provided same level of service. This class-based service provisioning is applied in DiffServ [BL98] that provides per-aggregate per hop behavior service.

2.6.2. Data Plane

Once a flow is accepted, the control plane would set the traffic policing and discarding policy at each node before passing control to the data plane. The functions in the data plane include: 1) Traffic policing; 2) Packet forwarding; 3) Buffering and discarding; and 4) Packet scheduling. The purpose of these functions is to protect real-time flows from QoS violation. The traffic policing function identifies and separates in-profile traffic from out-of-profile traffic. In general, a token bucket (TB) policer has an envelope $TB(\sigma, \rho)$, where σ represents the flow's burstiness and ρ the long term rate of the flow. The traffic bounded by this envelope constitutes the in-profile traffic otherwise; it is considered out-of-profile traffic. The scheduling mechanism would protect in-profile traffic; and treat out-of-profile traffic as part of the best effort traffic in the network. In the packet forwarding process, the identifiers of input packets are compared against the entries in the forwarding table updated by the control plane during the connection establishment. The integrity of this forwarding table at every node is a necessity for the correct operation of the delivery process. When a flow terminates, all of these tables would be updated to reflect the change of status.

The purpose of buffering at a node is to resolve the mismatch between the input rate and output rate at this node. This mismatch could happen in two possible scenarios. For the first scenario, the accumulation is due to the temporary overload of the link due to the temporal burstiness of the traffic. This could be easily resolved by simply providing a large enough buffer as this condition would disappear without further intervention. For the second scenario, input rate is consistently higher than the output rate due either to out-of-profile or best effort traffic. In this case, simple buffering is insufficient. The node has to discard appropriate packets when this congestion condition arises. We have discussed this issue in great details in Section 2.4. There are two ways of buffering at each node: input buffering or output buffering. In the former, packets are queued at the input buffer and released at appropriate time for service at the output link. In the latter, input packets are queued at the output buffer to wait for service. In the meantime, the packet scheduler would schedule backlogged packets according to its scheduling policy to the output link for service. For details in packet scheduling, the reader could refer to Sections 2.2 for a

detail review of common scheduling algorithms. It is worth noting that the time available for scheduling decisions would normally be very short and the scheduling algorithm must be very efficient to handle millions of flows for large bandwidth links. This is one of the main difficulties in the design of scheduling algorithm for QoS provisioning in a high-speed packet network.

2.7 Existing QoS Frameworks

2.7.1. Integrated Services (IntServ) [BR94]

The IntServ architecture is designed to provide per-flow QoS guarantees in the Internet paradigm. In best effort delivery, it does not provide any guarantee to users' traffic. A user would experience very good response time and throughput in idle periods of the Internet and long response time during periods of heavy utilization and no response in severe congestion. IntServ is an early approach to address the notion of QoS in the Internet. It is designed to provide strict service guarantee in which in-profile packets of a flow are protected and would be delivered within a delay bound. A shaper dimensioned according to the traffic parameters of the flow would regulate the traffic of a guaranteed flow. It is intended for applications that require deterministic QoS guarantees.

In most case, there is little attempt to minimize jitter but attention to bound the delay for each flow. Jitters are normally removed at a receiver by means of a playout buffer. This buffer should be large enough to cater to the maximum delay experienced by the flow such that the initial buffering prevents buffer starvation during playout. In most cases, packets arrived much earlier than this "maximum delay bound". For IntServ implementation, it assumes that every network element should be IntServ enabled for end-to-end guarantee. Each network element must ensure that the service provided approximates the theoretical "fluid model". This is equivalent to the service of a dedicated wire of bandwidth R between a source and a receiver.

The definition of guaranteed service is subjected to the arrival traffic shaped by a token bucket $TB(\sigma, \rho)$ with per node delay being bounded by σ / R for $R > \rho$, where R is the share of the bandwidth and ρ is the peak rate. The network element must ensure that the queueing delay of any packet be less than " $\sigma / R + C / R + D$ ", where C and D describe the maximal local deviation away from the fluid model. The error term C is the rate-dependent error term that accounts for the delay due to the rate parameters of a flow. Whereas, D is a rate-independent error term that represents the worst case delay per

element error due to non-rate based transit variations. This model assumes WFQ as the scheduling policy. If a flow is inactive for a short period of time, its unutilized share of service would be distributed to backlogged flows using WFQ without any memory for unutilized service.

In WFQ, the finishing time of every in-profile packet is computed to schedule packets in ascending order of their deadlines. This algorithm ensures that the output bandwidth is shared fairly amongst all backlogged flows proportional to their reserved rates. By reserving sufficient bandwidth for all flows sharing a link, it is shown in WFQ [PAR94] that every packet of the guaranteed flows could be serviced within its delay bound. To fulfill this condition, admission control function in IntServ at the ingress node utilizes the RSVP protocol to reserve the required bandwidth along the selected path of a flow. The DiffServ architecture is presented next.

2.7.2. Differentiated Services (DiffServ) [BL98]

DiffServ [BL98] defines a feasible architecture for implementing scalable service differentiation in the Internet. Service differentiation allows the provision of heterogeneous application requirements and permits differentiated pricing of Internet services. The architecture consists of the following functions in network nodes: 1) Packet classification; 2) Traffic conditioning that includes metering, marking, shaping and policing; and 3) Scheduling to implement per-hop forwarding behaviors. The network is divided into edge nodes and core nodes. Edge nodes are those with connections to external elements while core nodes have internal connections in a DiffServ domain, one that implements the DiffServ architecture. The classification, conditioning and scheduling functions are implemented at edge nodes while scheduling function is implemented at the core nodes. The basic idea in DiffServ is to keep core nodes simple and move the complexity to the edge nodes, which handle a small number of flows. While edge nodes can maintain per flow states and perform many functions, core nodes are stateless and perform simple scheduling, buffering and packet discarding functions.

DiffServ addresses the scalability issue by aggregation. While per flow state is maintain at edge nodes, each core node achieves scalability by applying per-hop behavior to aggregates of the traffic classified at the edge nodes. Without keeping any per-flow state at the core nodes, packets in a class but belonging to different flows would be treated the same manner independent of their original flows' identities. By maintaining a limited number of aggregates or classes, scalability is no longer an issue.

There are three types of service defined in the architecture: 1) Premium Service, 2) Assured Service and 3) Best Effort Service. The premium service defines a virtual lease line with a fixed maximum bandwidth. This is generally for the provisioning according to the peak capacity profiles. Traffic is shaped at the edge nodes and out-of-profile packets are dropped. The Assured Service defines a better than best effort service. The resources are provisioned statistically according to the expected capacity usage profiles. Traffic is policed at the edge nodes into in-profile and out-of-profile traffic. The former is "unlikely" to be dropped while out-of-profile traffic gets best effort service. This out-of-profile traffic forms part of the best effort traffic in the Internet. In [BL98, NI99], the Expedite Forwarding PHB and Assured Forwarding PHB are defined for the Premium and Assured Services respectively. We would be using these terms interchangeably in this review.

In order to provide the appropriate service to different flows, all flows are classified at ingress nodes for their packets to be forwarded according to their PHB. For multi-field classifications, packets could be classified according to the content of multiple fields in the IP header which could include 1) source and destination IP address, 2) source and destination port numbers, 3) protocol ID and 4) type of service (TOS) byte. These classified packets would form flows with different service requirements. Different flows are then aggregated together into different PHB aggregates and forwarded according to their aggregate's Per-Hop Forwarding Behavior (PHB). It specifies the external behavior when forwarding aggregates identified by the DSCP. This PHB defines specific requirements such as 1) QoS parameters including rate and delay and 2) how services are shared. These classifications affect how service is provided to different flows. As

discussed earlier, EF and AF are the two key PHB aggregates. The EF-PHB is designed to provide a low delay low loss service. This group of services includes telephone and video conferencing that require timely delivery of information but is loss tolerant. To give a low delay low loss example, consider the Voice-over-Internet-protocol (VoIP). Its end-to-end processes include signal compression, packetization, delivery and playout buffering before the voice is playout at the receiver. In general, an end-to-end delay of less than 100ms implies good voice quality even if small losses are encountered. In the case of video conferencing, an end-to-end delay of 600ms implies good video quality at the receiver. While the actual delay for subjective quality would differ from person-to-person, a specific delay bound would imply good quality reception of these loss tolerant applications. The coarse QoS guarantee in EF class works well for this group of traffic.

Assured Forwarding PHB is the other aggregates behavior defined for DiffServ. This PHB caters to a different group of applications that requires coarse rate guarantee and more tolerant to delay variation. Each of these flows requires a coarse rate guarantee and better than best effort service. This group of applications include video-on-demand, news casting and file transfer. A large playout buffer evens out the larger delay variation at the receiver. In general, the AF class could be subdivided into three subclasses: gold, silver and bronze and packets in these three sub-classes are scheduled using the WFQ discipline. The residual bandwidth after serving EF class of traffic is thus shared proportionally among the three sub-classes.

The BE delivery is provided to best-effort and out-of-profile traffic from the AF class of traffic discussed earlier. It utilizes spare capacity left over by the EF and AF classes. This service caters to a very large group of Internet users who are cost conscious but satisfied with a low cost best effort service.

From these discussions on DiffServ, we note that this architecture provides service differentiation by means of Behavior Aggregates. Static priorities are assigned to different PHB to implement the differentiated services. A decreasing order of priority is assigned to EF, AF and BE traffic, allowing services to be provided in this order. In

essence, DiffServ could provide a coarse QoS guarantee by the EF-PHB, a coarse rate guarantee for AF service together with a best effort service in the Internet.

2.8 Summary and Outstanding Issues

In this chapter, we have provided a review on packet scheduling, admission control, congestion control and two existing QoS architectures: Integrated Services and Differentiated Services. Scalability and computational complexity of a scheduling algorithm are two of the main issues that determine whether the algorithm is suitable for implementation. To find a simple and easy implementation algorithm that provides a range of deterministic QoS guarantees is still an open issue. For admission control, the problem is not as difficult as packet scheduling. Though it controls the admission of flow in real-time, it does not have the short time constraint as packet scheduling. The Service Curves theory has been applied successfully to derive the conditions for admission control. As the Next Generation Internet would carry best effort traffic besides guaranteed traffic, we provide a literature review on congestion control, highlighting the basic ideas in these research works. The climate of this review focus on two existing QoS architectures we mentioned at the beginning of this section. The review shows that there are gaps in the existing QoS architectures: 1) Integrated service, which is designed to provide deterministic guarantee, suffers from the lack of scalability; It would be suitable for a small network with thousands of flows but would not scale if millions of flow have to be processed; 2) Differentiated Service addresses the problem of scalability by means of flow aggregation but it could provide only a coarse QoS guarantees.

It is not obvious how all services could be integrated into the Next Generation Internet based on these existing architectures. In order to reap the maximum benefit from multiplexing gain, all services should be integrated in the NGI. Existing leased line users could enjoy a low cost of service while best effort users could enjoy greater capacity available at no additional cost. It seems that a range of per-flow QoS guarantees together with support for congestion control for best effort service would be desirable in the NGI. To this end, we revisit the issues of packet scheduling, admission control and congestion

control and propose solutions that address the issues of per-flow deterministic QoS guarantees and congestion control for best effort traffic in the next few chapters.

CHAPTER 3

A FLOW-STATE-DEPENDENT DYNAMIC PACKET SCHEDULING ALGORITHM FOR SERVICE CURVES ASSURANCE

The Service Curves model is an established method for per flow QoS provisioning. Due to its computational complexity in its Earliest Deadline First (EDF) policy and difficulties in tracking the curves in real time, little research work is available for practical implementation of service curves. In this chapter, we propose a novel packet scheduling algorithm, namely Flow-state-dependent Dynamic Packet Scheduling (FDPS) for a practical implementation of service curves. In this scheduling scheme, admitted QoS-based flows are protected yet allowing best effort traffic to co-exist in a multi-service network environment. The proposed FDPS scheme performs packet monitoring, marking, scheduling and discarding. With its fine granularity in packet marking, individual packets are forwarded and scheduled in a controlled and orderly manner. By means of mathematical analysis, we show that the FDPS algorithm realizes the arrival and service curves required in Service Curves and thus allows the theory of Service Curves to be applicable in a real network for per flow QoS provisioning. We also present an example for the admission control policing associated with FDPS. Compared with existing scheduling algorithms for QoS provisioning, FDPS achieves a tight end-to-end delay for QoS flows in a multi-services packet switching network. We conduct extensive simulation experiments based on NS-2 and the numerical results validate our theoretical analysis.

3.1 Introduction

The last decade has witnessed very rapid growth of the Internet accompanied by an evolution of new applications, ranging from complex applications including videoconferencing, IP telephony, interactive multimedia, etc, to simple data services. The former class of applications often require Quality of Service (QoS) guarantee, which cannot be provided by the current IP network architecture based on the best-effort service model. This new requirement calls for an architecture that can support QoS while preserving the scalability and simplicity of the current Internet.

Previous efforts to provision QoS in the Internet have resulted in the Integrated Service (IntServ) [BR94] and the Differentiated Services (DiffServ) [BL98] models. The purpose of IntServ is to provide per-flow absolute QoS guarantee. However, due to its poor scalability and a lagging demand for per-flow absolute QoS, the interest in Internet QoS eventually is shifted to the DiffServ architecture, which was proposed as a scalable solution based on the same paradigm as the Internet. This architecture employs simple packet scheduling and buffer management at the core routers based on tags that are set at the edge routers. The edge routers are allowed to perform traffic management on a per-flow basis while the core routers are not. In this architecture, the edge routers are responsible for the end-to-end service assurance. Although some traffic management and packet marking mechanisms have been proposed for the edge routers, whether DiffServ is able to deliver the end-to-end service differentiation or QoS guarantee across applications is not clear.

Over the past 15 years, new theoretical tools have been devised to analyze the performance properties of these algorithms. Multi-node delay bounds have been developed for networks of elements characterized by service curves using “network calculus” [CR91,CR91+], an approach which encompasses and generalizes previous results for networks of Weighted Fair Queueing (WFQ) servers [PAR94] and rate-controlled servers [ZH93]. In general, such techniques provide *schedulability conditions*, i.e., constraints that, if satisfied, ensure that all packets of all flows will meet their

respective delay bounds without violation or loss. Later development in Service Curves shows that this model could be used for end-to-end QoS provisioning.

In Service Curves, two curves are defined: (1) arrival curve, which specifies the traffic characteristics of a flow to be serviced; and (2) the minimum service curve, which specifies the lower bound of the service provided by a scheduler. The arrival curve is normally deduced from the traffic parameters of a flow specified in its Service Level Agreement (SLA) at the connection establishment. This curve defines the upper bound on the arrival traffic in which we consider as in-profile traffic or conformant traffic. Any traffic arrival above this upper bound is considered as out-of-profile or non-conformant traffic. Obviously, only conformant traffic is protected in the Service Curves method. The lower bound of service curve that is guaranteed to a flow is defined by the minimum service curve. The minimum service curve is a time-shift version of the arrival curve. Figure 3.1 shows the arrival curve $A(t)$ and minimum service curve $S(t)$, a time-shifted version of $A(t)$, i.e., $S(t) = A(t - d)$, where d is the maximum scheduling delay.

Figure 3.1 Arrival and service curves

In [CR95], Cruz *et.al* show that if the arrival and service curves are preserved at each node along a path of a flow, the QoS for that flow could be guaranteed. However, it is not obvious how the packets may be scheduled using the service curve directly. Two issues have prevented the implementation of service curves: 1) high computational complexity

of EDF algorithm based on the packet finishing time computed from each service curve; and 2) the difficulty in tracking the arrival and service curves in real-time. An emulator for realizing the arrival and service curves is needed for packet finishing time computation. The shifting of the service curve in space and time makes the tracking very computing intensive. Although there is much works [CR95, SAR95, CR91, CR91+] on Service Curves, to the best of our knowledge, there is little known work in providing a practical packet scheduling algorithm that satisfies the conditions of the Service Curves theory.

In this chapter, we present a novel Flow-state-dependent Dynamic Priority Scheduling (FDPS) to address this issue. We first prove that the FDPS algorithm realizes Service Curves to provide protection to QoS-based flows and it provides a seamless integration of QoS-based and best-effort traffic in a simple network. Next we would show that, by applying the theory of Service Curves, a number of cascaded network elements in a connection path could be transformed into three equivalent elements, namely, the equivalent arrival and service curve elements, and the equivalent delay element. To validate the mathematic analysis, we conduct extensive simulation experiments based on NS-2 to show that FDPS provides protection to QoS-based flows.

In satisfying the requirements of Service Curves for QoS provisioning, it is easy to model the end-to-end performance of the FDPS algorithm in the presence of uncertain traffic conditions in a network. In other words, the FDPS algorithm enables the interesting features of Service Curves to be realized. We also present an admission control example to demonstrate the effectiveness of the algorithm.

The rest of this chapter is organized as follows. We elaborate the FDPS algorithm and present an analysis of its operation in the next section. We prove that the FDPS realizes the Service Curves in Section 3.2. We discuss the properties and some implementation issues of FDPS in Section 3.3. We present the simulation results and discussions in Section 3.4. Finally, we conclude this chapter by highlighting our contributions in Section 3.5.

3.2 Flow-based Dynamic Scheduling (FDPS) Algorithm

In this section, we present a new scheduling algorithm, called Flow-state-based Dynamic Priority Scheduling Algorithm (FDPS). The algorithm is designed to provide per flow service protection. The net effect of such per flow protection is an end-to-end protected virtual channel for the flow. In most per node scheduling, only local information available at each node is utilized by the scheduling algorithm. Most algorithms do not consider whether a packet belongs to the in-profile or out-of-profile traffic, much less the extent of the conformance or non-conformance. Unlike these algorithms, the FDPS algorithm retrieves the state of a flow in relation to a packet to generate a dynamic priority for the purpose of scheduling. The state of a flow is then encapsulated in the dynamic priority assigned to a packet. Each of these packets is then forwarded to the output queues according to the corresponding priority before priority based scheduling is used to service these packets queued at the output buffer. Positive and negative priorities are used to represent the conformant and non-conformant states respectively. A simple priority scheduling determines which packet to be serviced at any moment in time. The dynamic priority represents the state of conformity or non-conformity of the packet in relation to the contractual traffic profile.

3.2.1 Conformant and Non-conformant Traffic in FDPS

The traffic on the Internet could be broadly classified as streaming and elastic traffic [KO04,YAS04]. Clear QoS requirements are normally specified for the former traffic while guarantees are not provided for the latter. To provide deterministic guaranteed for a streaming flow, its traffic profile needs to be specified in the SLA. Using this traffic profile, any traffic within their traffic profile is considered in-profile or conformant traffic. Otherwise, it is considered out-of-profile or non-conformant traffic. Packet arrivals from a real-time flow are constrained by arrival curves, which are described by general subadditive functions, each specifying the maximum amount of traffic that can arrive within any given time interval [CR91, CR91+]. For example, the $\sigma - \rho$ model is most commonly used to describe the worst-case traffic from a flow by a burst parameter σ and a long-term rate parameter ρ . As the notion of conformant and non-conformant

traffic represents important state of a flow in this chapter, we give the formal definitions as follows.

Definition 1 For the traffic arrival $A_i(t)$ of flow i , its conformant traffic is given by

$$\bar{A}_i(t) \leq \bar{S}_i(t) = \sigma_i + \rho_i t \quad (3.1)$$

where t is a time interval, σ_i is the depth of the leaky-bucket and ρ_i is the reserved rate for flow i . In this definition, conformant traffic is defined as the traffic under the envelope of leaky-bucket, which is defined as $\bar{S}_i(t) = LB(\sigma, \rho) = \sigma_i + \rho_i t$.

Definition 2 For the traffic arrival $A_i(t)$ of flow i , its non-conformant traffic is defined as

$$\tilde{A}_i(t) = \max \{0, A_i(t) - (\sigma_i + \rho_i t)\} \quad (3.2)$$

In this definition, the leaky-bucket operates as a traffic policer which identifies the non-conformant or out-of-profile traffic.

With the above definitions, it is clear to see that a QoS-based application may introduce both conformant traffic and non-conformant traffic to the network. The non-conformant part, which is identified and marked by the proposed FDPS algorithm, will be treated as best-effort at the router by the proposed scheme. In other words, the best-effort traffic in the discussions for the rest of the chapter may include the traffic from best-effort services and the non-conformant part of the traffic introduced by QoS-based applications.

3.2.2 The FDPS Algorithm

The FDPS algorithm performs traffic monitoring, priority marking and scheduling. Figure 3.2 shows the structure of the FDPS scheduler. An FDPS scheduler mainly consists of an Enhanced Token Bucket (ETB) which employs two token buckets, positive and negative buckets, for monitoring and marking packets and a number of output priority queues. The ETB does not regulate traffic but marks each arriving packet to forward it to the corresponding dynamic priority queue. Packets from each flow are monitored by the algorithm for its flow state (conformant or non-conformant) and marked accordingly. For each conformant packet a positive dynamic priority derived from the

ETB is assigned to the packet. The marking of the packet facilitates it to be forwarded to the corresponding dynamic priority queue at the output buffer. A total of p^+ output queues are available to conformant traffic and p^- output queues for non-conformant traffic. Note that p^+ and p^- also indicate the maximum and minimum dynamic priority assigned to a packet. At the output, a priority based scheduler services packets in the highest priority queue first.

Figure 3.2 Structure of an FDPS scheduler

The operation of the FDPS algorithm is given by the following pseudo-code.

FDPS algorithm:

Consider any output link for a given router. There are p^+ and p^- queues for conformant and best effort traffic respectively. $J = \{1, \dots, n\}$ is the set of n flows. $\bar{l}_i$ is the average packet length from flow i . $I_i^+(t)$ and $I_i^-(t)$ are the positive and negative dynamic priority corresponding to the number of tokens in the positive and negative buckets of flow i respectively. Upon the

arrival of the k^{th} packet of flow i , denoted by p_i^k , a corresponding dynamic priority q_i^k is computed and assigned to p_i^k

1) Initialize: at system start time t_0 ,

$$I_i^+(t_0) = p^+, I_i^-(t_0) = -1, \forall i \in J = \{1, \dots, n\};$$

2) Recalibrate:

$\forall i \in J = \{1, \dots, n\}$ after every time interval $\tau_i = \bar{I}_i / \rho_i$,

update $I_i^+(t)$ and $I_i^-(t)$ as follows:

$$I_i^+(t) = \min\{I_i^+(t-) + 1, p^+\};$$

$$I_i^-(t) = \min\{I_i^-(t-) + 1, -1\};$$

$$I_i^-(t) = \max\{I_i^-(t-) + 1, -p^-\};$$

3) Dynamic Priority Assignment:

upon the arrival of the k^{th} packet of flow i , p_i^k ,

If $I_i(t) > 0$

$$q_i^k \leftarrow I_i^+(t),$$

$$I_i^+(t) = \max\{I_i^+(t-) - 1, 0\}.$$

Else

$$q_i^k \leftarrow I_i^-(t),$$

$$I_i^-(t) = \max\{I_i^-(t-) - 1, -p^-\}.$$

Endif

Forward packet p_i^k to the output queue at
priority q_i^k

4) Packet transmission

The server serves the backlogged queue with the highest priority first.

The FDPS scheduler consists of two main processes: (1) the packet forwarding process; and (2) the priority scheduling process. The first process consists of an enhanced token-bucket (ETB) holding positive and negative tokens. Let the depths of the positive and negative token buckets be p^+ and p^- respectively, according to the number of the positive and negative output queues. These buckets are incremented every interval $\tau_i = \bar{l}_i / \rho_i$, where $\bar{l}_i$ is the mean packet size and ρ_i is the long term rate (reserved rate) of the flow. Upon arrival of a packet, the ETB is checked and a dynamic priority is assigned to this packet. All forwarding processes are independent and incoming packets from different flows are processed simultaneously. Therefore, the FDPS algorithm utilizes the flow state of a packet to generate a dynamic priority to forward a packet to its output queue. This is different from the rate control static priority (RCSP) scheduling [ZH93] in which all packets of a flow are forwarded to the same priority queue once they are released by the rate/delay regulator. The FDPS algorithm does not control the rate of a flow but monitors its conformance status and assigns a dynamic priority to forward a packet to a corresponding output queue and the packets in the highest priority queue are serviced first.

In the FDPS implementation, connection admission control is activated when a new connection request arrives. A request for guaranteed service is admitted only if there are sufficient network resources to support this request. The traffic parameters of a flow given by the traffic profile could be represented by a triple (σ_i, ρ_i, d_i) , where σ_i and ρ_i represent the depth and long-term rate of the flow and d_i is the maximum end-to-end delay. If the traffic arrival is within the envelope $LB(\sigma_i, \rho_i)$ of this token bucket, all packets are marked with positive dynamic priorities. Non-conformant traffic is thus marked with negative dynamic priorities. To ensure that the traffic from a source to be conformant, it may be smoothed by a regulator with envelope $LB(\sigma_i, \rho_i)$ at the source. The issue of out-of-sequence packets at the destination could be handled by upper layer functions.

In FDPS, the non-conformant packets are forwarded to the corresponding negative priority queues. When a negative bucket is empty, the assigned dynamic priority is $-p^-$. Thus, non-conformant packets of an aggressive flow would have packets marked with $-p^-$ priority. Also, a new non-conformant flow would have a maximum $(p^- - 1)$ packet precedence over packets of a very aggressive non-conformant flow for which its negative bucket is empty.

3.2.3 Operation of FDPS and analysis

In this section, we detail the operation of the FDPS scheme and illustrate how it protects QoS-based flows and yet allows best effort traffic to co-exist in the same packet switching network. The key feature of the FDPS algorithm is the enhanced token bucket (ETB) that uses two buckets: positive and negative. As the negative dynamic priority assignment is derived from the latter, we focus our attention on the positive dynamic priority assignment process. Figure 3 shows the positive and negative buckets of an ETB. Instead of traffic regulation for a normal Token Bucket, the purpose of this ETB is to mark each packet with a dynamic priority according to the state of its flow. We proceed to analyze how this dynamic priority is derived. We assume that the traffic profile of a QoS-based flow i is specified by a leaky-bucket envelope $LB(\sigma_i, \rho_i)$, where σ_i is the depth of the bucket and ρ_i the long-term service rate.

Figure 3.3 Schematic diagram of the Enhanced Token Bucket.

Figure 3.3 shows a schematic diagram of the Enhanced Token Bucket (ETB) in which both the positive and negative bucket are filled at the rate ρ_i for flow i . Similar to Figure 3.2, p^+ and p^- are the depths of the positive and negative bucket respectively. Y_i^+ and Y_i^- represent the number of tokens in the positive and negative buckets, and P_i^k represents the packet k arriving at the input buffer. Without any loss of generality, we assume that the flow i uses fixed packet size l_i and a token is added to each bucket at every interval l_i / ρ_i . In Figure 3.3,

$$Y_i = \begin{cases} Y_i^+, & \text{if } Y_i^+ > 0 \\ Y_i^-, & \text{otherwise} \end{cases}$$

A switch is provided in the diagram that switches from the positive to negative bucket when $Y_i^+ = 0$. The following proposition is prescribed to state how a dynamic priority is assigned to a packet.

Proposition 3.1 The priority q_i^k assigned to packet k in the input buffer is given by

$$q_i^k = f(P_i^k, Y_i)$$

where

$$f(P_i^k, Y_i) = \begin{cases} Y_i^+ & \text{if } Y_i^+ > 0 \\ -(p^- + 1 - Y_i^-) & \text{otherwise} \end{cases}$$

To prove the above proposition, we analyze the procedure in the FDPS algorithm in three different cases in the following.

Case 1: Suppose the first packet of a flow arrives at a full positive bucket, we have $Y_i^+ = p^+$, $Y_i^- = p^-$ and $k = 1$.

The assigned priority q_i^k for $k = 1$ is

$$q_i^1 = Y_i^+ = p^+ \quad \text{for } Y_i^+ > 0$$

Subsequent packets before any token re-filling will be assigned $q_i^2 = p^+ - 1, q_i^3 = p^+ - 2$, etc.

For example, let us consider 3 packets arrived at the input buffer as a burst when the buckets are full. For the third packet ($k = 3$) the assigned priority is

$$q_i^3 = Y_i^+ - k + 1 \quad \text{for } Y_i^+ \geq k$$

For $Y_i^+ = p^+, k = 3$

$$q_i^3 = p^+ - 3 + 1 = p^+ - 2$$

This is the same as the result deduced earlier.

Case 2: Suppose a packet arrives when there are Y_i^+ tokens in the buffer. In this case, we could set $k = 1$ and $q_i^k = Y_i^+ - 1 + 1 = Y_i^+$. It is interesting to note that a priority assigned to a new arriving packet is identical to the number of tokens in the positive bucket when $Y_i^+ > 0$. In Figure 2.2, $I_i^+ = \max\{1, Y_i^+\}$ for $0 < Y_i^+ \leq p^+$. Hence we have $q_i^k = I_i^+$ for $Y_i^+ > 0$. Clearly, this is the case when the positive bucket is non-empty. As a result, I_i^+ could be considered as a pointer for priority assignment when $Y_i^+ > 0$.

Case 3: Consider the case in which $Y_i^+ = 0$ and packet k arrives. Note that this is the case of non-conformant flow in which the arrival traffic exceeds the envelope of the positive token bucket. Now the switch in Figure 2.3 is latched to the negative bucket. The priority assigned to this packet k is given by $q_i^k = -(p^- + 1 - Y_i^-)$ for $Y_i^+ = 0$. Note that $I_i^- = \max\{-p^-, (p^- + 1 - Y_i^-)\}$, bounding I_i^- between $-p^-$ and -1 . The first packet k arrival after the positive bucket is empty ($Y_i^+ = 0$) sets $q_i^k = -1$ as $Y_i^- = p^-$ for a full negative bucket. Subsequent packet arrivals before any token re-filling causes priorities $-2, -3, -4, \dots$ etc, to be assigned. Hence I_i^- could be considered as a pointer on the negative bucket with -1 and $-p^-$ as the top and bottom of the negative bucket. In general, the priority assigned to packet k when $Y_i^+ = 0$ is $q_i^k = I_i^-$.

We have discussed the procedure of the priority assignment. Here let us discuss the design of the two token buckets in the ETB in more detail. If a single bucket of depth $(p^+ + p^-)$ is used for priority generation, there is a “latch on” effect in the dynamic priority

assignment mechanism. Once a flow becomes non-conformant and the “negative” region of the bucket is used, the priority would be a negative value—unless the flow stops transmission and allows the refilling process to increase the number of tokens in the bucket and brings the tokens into the “positive” region. This problem is rectified by having two token buckets refilled at the long-term rate ρ_i for flow i . If all traffic belonging to a flow is conformant, only the tokens in the positive token bucket are used for priority assignment.

In the following, we further investigate the properties of the priority assignment in the FDPS in more details.

Lemma 3.1 The dynamic priority q_i^k assigned to packet k of a conformant flow i is always positive. That is,

$$q_i^k = f(Y_i, X_i) = I_i^+(t) \geq 1 \quad (3.3)$$

This follows directly as the traffic profile of a flow and the positive ETB envelope are identical.

For a given traffic profile of a flow, it may be more convenient for the source to do some preprocessing of its traffic to ensure that all “must reach” information is transmitted as conformant traffic while “good to reach” information is marked as non-conformant traffic as part of its best effort traffic. All ‘must reach’ information could be shaped by a leaky-bucket $LB(\sigma_i, \rho_i)$ regulator as conformant traffic while “good to reach” packets are marked as best effort traffic before being injected into the network. A bit could be used to identify whether a packet is a part of the best effort traffic transmitted by a source. We assume that a non-conformant bit (a bit of the IPv4 TOS octet or IPv6 traffic class octet) may be used to indicate a best-effort packet. Using this indicator, conformant or non-conformant packets could be transmitted in an interleaved manner yet allowing the FDPS algorithm to identify the non-conformant packets via this bit. The FDPS mechanism

would assign the appropriate dynamic priority for the scheduling. We state the following Lemma for non-conformant traffic.

Lemma 3.2: A non-conformant packet, say packet P_i^k of flow i , is always marked with a negative dynamic priority q_i^k by the FDPS algorithm. That is,

$$q_i^k = f(P_i^k, Y_i) = I_i^- \leq -1 \quad (3.4)$$

As the positive token bucket is empty when non-conformant packets arrive, the negative token bucket is used for priority assignment. For packets with non-conformant bit set by the source, the negative bucket is accessed for priority assignment. The result follows.

The ETB enables clear identification of whether a packet is conformant (in-profile) in relation to its leaky-bucket envelope $LB(\sigma_i, \rho_i)$. Every packet is marked with a dynamic priority according to the state of its flow. The mapping of the state of a flow to a dynamic priority using the enhanced token bucket provides the fine granularity for packet scheduling at low computational complexity. The above analysis shows that the computation of the dynamic priority is independent of the very large number of flows in a high bandwidth link.

We have assumed that all flows use fixed packet sizes before for the ease of analysis. In practical implementation, this assumption can be released. We recall that the token refilling procedure uses a mean packet size $\bar{l}_i$ for the refilling interval, i.e., $\tau_i = \bar{l}_i / \rho_i$ for both the positive and negative token buckets. For a packet of length l_i arriving from flow i , the priority assignment procedure is modified as:

Case 1: for conformant traffic

$$f(P_i^k, Y_i) = I_i^+$$

and

$$Y_i^+(t) = I_i^+(t) - \lceil l_i / \bar{l}_i \rceil$$

where $\lceil x \rceil$ gives the lowest integer greater than the argument x .

Case2: for non-conformant traffic

$$f(I_i, X_i) = \max\{p^-, -(1 + p^-) + Y^-(t)\}$$

$$Y_i^-(t) = Y_i^-(t) - \lceil l_i / \bar{l}_i \rceil$$

Note that there is some approximation due to the different packet sizes transmitted by a flow. In addition, a large packet arrival when the positive token bucket has only one token leads to marking this packet with priority “1” while part of the packet would constitute non-conformant traffic. Due to over provisioning in FDPS to meet delay requirements of all QoS-based flows and the low probability that all flows bursting together as investigated in large deviation theory [SHW94], it is reasonable to assume that QoS violation due to variable packet size could be neglected.

3.2.4 Operation Example

We elaborate the operation of FDPS through an example in this sub-section. We start the discussion by investigating the difference between the FDPS and the Weighted Fair Queuing (WFQ) algorithms [PAR93, PAR94]. For the sake of simplicity, we consider four flows f_1, f_2, f_3 and f_4 with fixed packet size l sharing a common link. Their traffic envelopes are assumed to be $LB(7l, \rho)$, $LB(5l, \rho)$, $LB(3l, \rho)$ and $LB(l, \rho)$ respectively. Three cases are used to illustrate how the FDPS operates.

Case 1: Flows f_1, f_2, f_3 and f_4 arrive at the node simultaneously. If all flows transmit with maximum burstness, the dynamic priority assigned to each packet of every flow by the FDPS algorithm by means of the ETB as shown in Figure 3.2 is given below.

flow	dynamic priority assignment
f_1	7, 6, 5, 4, 3, 2, 1, 1...
f_2	5, 4, 3, 2, 1, 1...
f_3	3, 2, 1, 1...
f_4	1, 1...

Each packet is then forwarded to the corresponding output buffer with the same priority as shown in Figure 3.2. The priority scheduler then schedules the packets in the highest priority queue for service. If we use P_i^k to represent a packet with dynamic priority k of flow i , the service sequence is given below:

In this sequence, note that bursty packets of a flow are serviced before the constant rate packets of other flows. It is easy to recognize that there are three distinct service periods. In the first distinct period, only one flow is serviced. In this example, 2 packets P_1^7 and P_1^6 for flow f_1 are serviced before the scheduling enters a second period. The second period could be subdivided into many sub-periods with different number of flows being serviced. In this example, flows f_1 and f_2 are serviced followed by flows f_1, f_2 and f_3 together. If more flows are using this link, this period may span many sub-periods with different number of flows be serviced. The third distinct period is entered when all flows are transmitting at constant rates. Packets in this period are scheduled in a first-come-first-served manner.

For a better understanding of the FDPS operation, we would give more details of the above-mentioned three distinct periods, called dominant, dominant weighted fair queueing (WFQ) and FCFS periods.

- (1) Dominant period: This period is defined as an interval in which only one flows is serviced due to the dynamic priorities of its packets assigned by the FDPS algorithm. In case 1, only flow 1 is serviced during this period due to priorities 7 and 6 assigned to the first two packets. In general, this period is very short as there is normally a limited number of packets with such high priorities and the state of the link changes to the second period.

- (2) Dominant Weighted Fair Queueing period: This period is defined as the interval in which a few (two or more) of the backlogged flows are serviced. As the FDPS algorithm scheduled the packets in the highest priority queue first, only flows with packets at this priority are serviced. After these packets are serviced, packets of the next highest priority would be scheduled unless packets of higher priority arrive. This scheduling sequence is repeated allowing the priority being serviced to decrease. This period could be divided into multiple periods of weighted fair queueing in which different number of backlogged flows are serviced. Two distinct dominant WFQ periods can be identified in case 1 in which flows f_1 and f_2 , or flows f_1 , f_2 and f_3 are serviced. When the burstiness of all flows have dissipated, the link enter into the third period.
- (3) First-come-first-served period: This period is defined as the interval in which the burstiness of all QoS flows have dissipated. These flows are now transmitting at constant rate and assigned a priority "1". As all packets have the same priority, packets are queue for service according to their order of arrival leading to the FCFS scheduling.

Case 2: Flows f_1 and f_2 arrive earlier and their burstiness has dissipated before flow f_3 arrives. As flows f_1 and f_2 are transmitting at the constant rates when flow f_3 arrives, the service sequence according to the priority assigned by FDPS in Figure 3.2 is

$$\underbrace{P_3^3, P_3^2, P_1^1, P_2^1, P_3^1, P_1^1, P_2^1, P_3^1, \dots}_{\text{Dominant period}} \quad \underbrace{\hspace{10em}}_{\text{FCFS period}}$$

We notice that two packets from flow f_3 are serviced in the dominant period before the FCFS period is entered. In FDPS, the bursty conformant packets are always serviced before the constant rate packets. These bursty packets are part of the conformant traffic of flow 3. This is different from the WFQ algorithm.

In the above discussions, we have also illustrated the subtle difference between WFQ [PAR93, PAR94] and FDPS algorithms. In WFQ scheduling, the aim is to share the

output link capacity among all backlogged flows according to their assigned weights. Whereas, the FDPS algorithm recovers the state of a flow to assign a priority to each packet which determines its order of service, highlighting the difference between the two scheduling algorithms.

It is interesting to discuss a scenario where a flow, say flow i , with bursty traffic parameters $LB(\sigma_i, \rho_i)$ transmits at only long-term rate ρ_i . Then each of its packets is assigned with highest dynamic priority p^+ by the FDPS algorithm. In other words, this flow would be given high priority service leading to a very low scheduling delay bound. It seems this is unfair to other QoS flows. If we investigate it carefully, we will find it actually has a positive impact on the scheduling delay bounds on other flows if Theorem 3 is satisfied, as none of the packets from other flows is delayed by the bursty packets of this flow leading to a lower delay bound. Obviously, the end-to-end delay of such a flow is much lower than the rest of the flows. This problem is due to the discrepancy between the traffic parameters and actual traffic.

3.3 Proof of FDPS Realization of Service Curves

The basis for QoS provisioning in Service Curves rests on its specification of the arrival and service curves for each flow. If and only if the FDPS algorithm fulfills these requirements, its performance is bounded by Service Curves. In this section, we will show how FDPS realizes Service Curves and thus provides per-flow QoS guarantees.

Proposition 3.2 The FDPS algorithm identifies conformant packets specified by the arrival curve $\hat{\lambda}_i(t)$ of the SLA by marking them with positive dynamic priority.

Proof: Note that the traffic specification of a flow is given in the SLA when a QoS-based connection request is made. If the traffic specification of a flow is given in terms of Leaky Bucket $\hat{\lambda}_i(t) = \sigma_i + \rho_i t$, the ETB could be equivalently parametrised. As a result, all positively marked packets by the algorithm constitutes in-profile traffic. If the flow becomes more bursty than this specification, the positive token bucket would become

empty and the negative bucket is accessed for dynamic priority assignment. Such out-of-profile packets would be assigned negative priority. Hence, this method of priority assignment ensures that all in-profile traffic defined by the arrival curve is assigned positive priority.

As positive priority packets are serviced before negative ones, there is clear isolation between in-profile and out-of-profile traffic. Therefore, the FDPS algorithm marks all in-profile traffic with positive priority and this completes the proof.

Having all in-profile packets marked by positive priorities, we have to prove that the minimum service curve is guaranteed. In order to provide a minimum service curve to a flow, there must be sufficient capacity at a shared link to service all flows and the packets of this flow must be scheduled in such a way that service is received is lower bounded by its minimum service curve.

Proposition 3.3 The minimum service curve for flow i at link j is guaranteed by the FDPS algorithm if and only if the service curve for flow i is given by

$$S_i(t) = \hat{A}(t - d_j^{s-\min}) \quad \forall i \text{ at link } j$$

where $d_j^{s-\min} = \min_{\forall k} \{d_k^s\}$ is the minimum scheduling delay at link j and

$$\sum_{k=1}^n S_k(t) + L_{\max} \leq C_j t \quad \text{for } j \in \text{Path}_i$$

where $L_{\max}$ is the maximum packet size, C_j is the link capacity of link j , Path_i is the set of all links in the path of flow i and n is the number of flows sharing link j .

Proof: Suppose the arrival curve is given by $\hat{A}_i(t) = \sigma_i + \rho_i t$, the minimum service curve is given by $\hat{A}_i(t - d_i^s)$ where d_i^s is the maximum scheduling delay. It represents the lower bound of service needed. Obviously, a service curve $S_i(t) \geq \hat{A}_i(t - d_i^s)$ would be more than sufficient to provide the guaranteed service.

To ensure that there is sufficient capacity at any link j , for n QoS-based flows number from 1 to n , we require

$$\sum_{k=1}^n S_k(t) \leq C_j t$$

In other words, the sum of all the n service curves is always less than the service envelope of the link. For preemptive scheduling, we modify this inequality as

$$\sum_{k=1}^n S_k(t) + L_{\max} \leq C_j t$$

where $L_{\max}$ is the maximum packet size for link j . With this condition, we ensure that sufficient capacity is available at each link j . This condition is needed at each link in the path of a flow.

Figure 3.4 Two sets of arrival and service curves.

Let us consider the arrival and service curves of two flows, i and l at a shared link, as shown in Figure 3.4. Note that the maximum scheduling delays d_i^s and d_l^s of flows i and l are different. If both flows arrive together, the bursty packets from flow l would interfere with flow i . As the FDPS algorithm schedules packets in the highest priority queue first (Figure 3.2) and to meet the scheduling delay of both flows, the allocated aggregate

service curve is the sum of the arrival curves of flows i and l shifted by delay d_i^s , resulting in the following service curves:

$$S_i(t) = \hat{A}_i(t - d_i^s)$$

and

$$S_l(t) = \hat{A}_l(t - d_l^s)$$

where $d_i^s < d_l^s$. This reduces the scheduling delay bound of flow l to d_i^s ($< d_l^s$). With this implementation, all positive priority packets of both flows are serviced by time d_i^s . Non-bursty packets from flow l are also serviced within this delay bound. Such an implementation would ensure that the minimum service curves $\hat{A}_i(t - d_i)$ and $\hat{A}_l(t - d_l)$ are both satisfied due to

$$\hat{A}_i(t - d_i) + \hat{A}_l(t - d_l) < S_i(t) + S_l(t)$$

For n flows sharing a link j , we set

$$S_k(t) = \hat{A}_k(t - d_j^{s-\min}) \quad \text{for } k = 1, 2, \dots, n$$

and

$$d_j^{s-\min} = \min_{\forall k} \{d_k^s\}$$

All conformant packets are serviced by $d_j^{s-\min}$ at link j . As the worst case when all n flows burst together is satisfied, delay is not an issue in other periods of the flows. Hence, we have proved the FDPS is able to guarantee the minimum service curve for each flow.

From propositions 3.2 and 3.3, it is recognized that admission control at an ingress node is needed for the FDPS algorithm. Unless there is sufficient capacity to support a new request, it would be rejected. This proposition is required at every link along the path of a new flow. Capacity reservation is needed to ensure that

$$\sum_{k=1}^n S_k(t) + L_{\max} \leq C_j t \quad \text{for } j \in \text{Path}_i$$

where $Path_i$ represents the set of links in the path of new flow i . We will further discuss the admission control policy associated with FDPS through an example in more detail in the next section.

From these discussions, we state the following theorem.

Theorem 1 The FDPS algorithm realizes Service Curves if and only if the admission control function ensures that

$$\sum_{k=1}^n S_k(t) + L_{\max} \leq C_j t \quad \text{for } j \in Path_i$$

$$S_k(t) = \hat{A}_k(t - d_j^{s-\min})$$

where $d_j^{s-\min} = \min_k \{d_k^s\}$ is the minimum scheduling delay at link j with n flows and the ETB implements the arrival curve $\hat{A}_k(t)$ for flow k at link j .

Proof: this follows directly from Propositions 3.1 and 3.2.

3.4 Analysis and Properties of FDPS

3.4.1. Analysis of FDPS using Service Curve

In the previous section, we have shown how the FDPS algorithm realizes Service Curves [CR95, CR98, AG99] and thus the theories that apply to Service Curves are applicable to the FDPS algorithm. To provide performance bound for the FDPS algorithm, we apply the theory of service curve and model a connection as a series of network elements cascaded together to form an end-to-end connection. The theory of service curve is then applied to deduce the equivalent service elements that would characterize the conclusion, independent of traffic conditions.

Figure 3.5 shows the end-to-end network elements of a QoS-based flow. In this figure, $\bar{S}_j$, S_j and d_j represent the shaper envelope, service curve and delay elements for link j , with a total of V links in the path. By means of network calculus, this collection of

network elements would be reduced to three elements, namely, an equivalent arrival curve, an equivalent service curve and an equivalent delay element.

Figure 3.5 A series of network elements of flow i .

As QoS guarantee applies only to conformant traffic, we assume that the traffic of flow i transmitted by a source is shaped by a regulator with envelope $\bar{A}_i(t)$ and in particular, $\bar{S}_m(t) = \hat{A}_i(t)$ for $m = 1, 2, \dots, v$ in Figure 3.5 indicating that the traffic from the source is conformant.

Lemma 3.3 For a flow passing through a series of V shapers and V service curves, the end-to-end service of elements could be reduced to a shaper element $\bar{S}_{1V}(t)$ and a service curve element $S_{1V}(t)$ given by

$$\bar{S}_{1V}(t) = \bar{S}_1(t) * \bar{S}_2(t) * \dots * \bar{S}_V(t)$$

$$S_{1V}(t) = S_1(t) * S_2(t) * \dots * S_V(t)$$

Proof: For the case for two nodes, there are two shapers and two service curves as shown in Figure 3.6.

Figure 3.6 The equivalent shaper envelope and service curve.

Applying min-plus calculus [CR95, BOU01] to Figure 3.6, the output traffic

$$\begin{aligned}
 A_3(t) &= A_1(t) * \bar{S}_1(t) * S_1(t) * \bar{S}_2(t) * S_2(t) \\
 &= A_1(t) * \bar{S}_1(t) * \bar{S}_2(t) * S_1(t) * S_2(t) \\
 &= A_1(t) * \bar{S}_{12}(t) * S_{12}(t)
 \end{aligned}$$

where $\bar{S}_{12}(t) = \bar{S}_1(t) * \bar{S}_2(t)$ and $S_{12}(t) = S_1(t) * S_2(t)$. By induction, the result follows.

For the fixed delay in all end-to-end connection, it is desirable to model it as an equivalent delay element [AG99]. The following Lemma could be used to compute the equivalent delay.

Lemma 3.4. If a flow passes through v delay elements $\delta_{d_1}(t), \delta_{d_2}(t), \dots, \delta_{d_v}(t)$, the equivalent delays is given by

$$\delta_{d_{12}}(t) = \delta_{d_1}(t) * \delta_{d_2}(t) * \dots * \delta_{d_v}(t)$$

where

$$\delta_d(t) = \begin{cases} 0 & \text{for } t < d \\ \infty & \text{else} \end{cases}$$

Proof: Suppose the traffic $A_1(t)$ of a flow passes through two delay elements with delays d_1 and d_2 respectively. Using $A(t) * \delta_d(t) = A(t-d)$ [AG99] for the departure process for two elements is given by

$$\begin{aligned} A_3(t) &= A_1(t) * \delta_{d_1}(t) * \delta_{d_2}(t) \\ &= A_1(t-d_1) * \delta_{d_2}(t) \\ &= A_1(t-d_1-d_2) \\ &= A_1(t) * \delta_{d_{12}}(t) \end{aligned}$$

where

$$\delta_{d_{12}}(t) = \delta_{d_1}(t) * \delta_{d_2}(t)$$

Intuitively, the delay of the equivalent delay element is the sum of the delays of the two elements in series. By deduction, the result is extended to a series of V delay elements:

$$\delta_{d_{1V}}(t) = \delta_1(t) * \delta_2(t) * \dots * \delta_V(t)$$

This completes the proof.

An end-to-end connection may be represented as a concatenation of various elements as shown in Figure 3.5. We propose the next theorem which computes the equivalent elements that represent the connection. This leads to a model which is very useful in admission control in the next section.

Theorem 3.2 For a series of V shapers, V service curves and V delay elements along the path of a flow, the series of elements could be represented by an equivalent shaper element $\bar{S}_{1V}$, an equivalent service curve element S_{1V} and an equivalent delay element $\delta_{d_{1V}}$.

For flow i passing through $\bar{S}_1(t), S_1(t), \delta_{d_1}(t), \dots, \bar{S}_V(t), S_V(t), \delta_{d_V}(t)$, where $\bar{S}_j(t)$ is the envelope of the shaper at node j , $S_j(t)$ is its service curve and $\delta_{d_j}(t)$ is its delay element.

The three equivalent elements are given by

$$\begin{aligned} \bar{S}_{1V}(t) &= \bar{S}_1(t) * \bar{S}_2(t) * \dots * \bar{S}_V(t) \\ S_{1V}(t) &= S_1(t) * S_2(t) * \dots * S_V(t) \\ \delta_{d_{1V}}(t) &= \delta_{d_1}(t) * \delta_{d_2}(t) * \dots * \delta_{d_V}(t) \end{aligned}$$

Proof: Applying Lemmas 3 and 4, the output traffic $A_{V+1}(t)$ after passing through all these elements is given by

$$\begin{aligned} A_{V+1}(t) &= A_1(t) * \bar{S}_1(t) * S_1(t) * \delta_{d_1}(t) * \dots * \bar{S}_V(t) * S_V(t) * \delta_{d_V}(t) \\ &= A_1(t) * \bar{S}_1(t) * \bar{S}_2(t) \dots * \bar{S}_V(t) * S_1(t) * S_2(t) \dots * S_V(t) * \delta_{d_1}(t) * \delta_{d_2}(t) \dots * \delta_{d_V}(t) \\ &= A_1 * \bar{S}_{1V}(t) * S_{1V}(t) * \delta_{d_{1V}}(t) \end{aligned}$$

The result follows.

We have reduced the series of V shapers, V service curves and V delay elements to three equivalent elements $\bar{S}_{1V}(t)$, $S_{1V}(t)$ and $\delta_{d_{1V}}(t)$, which would be very useful for the analysis and admission control. Note that once these elements are obtained, we could apply FDPS algorithms to schedule the packets in conformance to Service Curves. As Serves Curves could provided per flow QoS guarantee independent of other traffic, the FDPS algorithm is able to provide per-flow QoS independent of the behaviors of other traffic iff Theorems 1 and 2 are applied. We would show how this is applied in admission control in the next section.

3.4.2. Admission Control for FDPS

In section 3.4.1, we apply the theory of Service Curves to reduce a series of network elements to three equivalent elements. According to Theorem 3.2, we could represent an end-to-end connection by the equivalent shaper, service curve and delay elements for flow i . With many flows sharing an output link, we propose the following theorem, which ensure Service Curves conformance for all flows.

Theorem 3.3 Given $\hat{A}_i = \sigma_i + \rho_i t$ as the envelope of the in-profile traffic of flow i , and C_j the capacity of link j in its path, this flow i is admitted iff

$$C_j \geq \frac{\sum_{\forall k} \sigma_k + L_{\max}}{d_i^{s-\min}} \quad \text{for } \forall j \in Path_i$$

where $L_{\max}$ is the maximum packet size at link j and $Path_i$ represents the set of links in the path of flow i , and

$$d_j^{s-\min} = \min\{d_k^s\} \quad \forall k \text{ at link } j$$

for the QoS of every flow to be guaranteed by the FDPS algorithm.

Proof: All conformant packets are protected against non-conformant packets by the FDPS algorithm due to the way priorities are assigned. We recall that in the discussion on the operation of the FDPS algorithm in Section 3.2, packets from different flows are interleaved in the service order according to the dynamic priority assigned to each packet. Due to the way dynamic priority is assigned, all conformant bursty packets are serviced before constant rate packets. If the arrival envelope of flow i is given by the leaky bucket envelope $LB(\sigma_i, \rho_i)$, the worst case of this bursty traffic is

$$\begin{aligned} \sum_{\forall k} \hat{A}_k(t) &= \sum_{\forall k} (\sigma_k + \rho_k t) \quad \text{for all flows sharing link } j \\ \therefore \sum_{\forall k} \hat{A}_k(0) &= \sum_{\forall k} \sigma_k \end{aligned} \quad (3.5)$$

Note $\hat{A}_k(t)$ is the arrival envelope of flow k and $\bar{S}_m(t)$ and $S_m(t)$ are shaper envelope and service curve for link m . Using Proposition 3.3 and noting $\bar{S}_m(t) = \hat{A}_i(t)$ for $\forall m \in Path_i$ for the positive token bucket in Figure 3.2, we have

$$\sum_{\forall k} \bar{S}_k(t - d_j^{s-\min}) + L_{\max} \leq C_j t \quad \forall j \in Path_i \quad (3.6)$$

Then

$$C_j \geq \frac{\sum_{\forall k} \sigma_k + L_{\max}}{d_j^{s-\min}} \quad (3.7)$$

where

$$d_j^{s-\min} = \min\{d_k^s\} \quad \forall k$$

This completes the proof.

Using this condition, the delay requirement of every flow is satisfied. In contrast to Diffserv that limits the expedited forwarding service to a fraction of the link capacity, the FDPS algorithm releases this assumption but subject to inequality (3.7). Theorem 3.3 ensures that the QoS of all flows sharing a common link are satisfied. However, it requires some over provisioning due to the minimum scheduling delay $d_j^{s-\min}$. This issue is illustrated in the following example.

3.4.3 Admission Control Example

Admission control is an important issue in our QoS provisioning mechanism. To illustrate how admission control can be implemented in FDPS, we have provided an example in which a flow is admitted only if the capacity at every link, along the path of a connection, is greater than the aggregate flow requirement, to prevent delay bound violation.

Suppose two flows x and y with parameters (σ_x, ρ_x, d_x) and (σ_y, ρ_y, d_y) , where σ_i and ρ_i are the depth and long-term rate of the leaky bucket and d_i is the delay bound of flow i , share a connection link j and their in-profile traffic are represented by $\hat{A}_x(t)$ and $\hat{A}_y(t)$ respectively. Therefore, $\hat{A}_x(t) = \sigma_x + \rho_x t$ and $\hat{A}_y(t) = \sigma_y + \rho_y t$.

Let d_i be the end-to-end delay bound for flow i . It may be divided into two components. A fixed delay d_i^f and a variable delay d_i^v . The fixed delay is due to the transmission and propagation delay of the links along the path of a flow. For a network using the FDPS algorithm, the variable delay is equal to the scheduling delay d_i^s , i.e., $d_i^v = d_i^s$. Therefore, the allowable scheduling delay $d_i^s = d_i - d_i^f$.

Assuming $\hat{A}_i(t) = \sigma_i + \rho_i t$ is the arrival curve and letting $\bar{S}_i(t) = \hat{A}_i(t)$, the minimum service curve is

$$S_i(t) = \bar{S}_i(t - d_i^s) \quad (3.8)$$

If rate r_i is allocated to flow i at this link, $S_i(t) \leq r_i t$ is required.

Substituting $S_i(t) \leq r_i t$ into equation (3.8), we get

$$\bar{S}_i(t - d_i^s) \leq r_i t$$

Recognizing $r_i \geq \rho_i$, let $t = d_i$ to find the lower bound of r_i , we have

$$r_i \geq \frac{\sigma_i}{d_i^s}$$

Figure 3.7 Service curves for flows x and y.

Figure 3.7 shows the service curves for flow x and y if the two flows are independent:

$$S_x(t) = r_x t \geq \frac{\sigma_x}{d_x^s} t, \quad r_x \geq \rho_x \quad (3.9)$$

$$S_y(t) = r_y t \geq \frac{\sigma_y}{d_y^s} t, \quad r_y \geq \rho_y \quad (3.10)$$

In view of the interaction between the two flows, we apply Theorem 3.3 to these flows. Under the assumption of $d_x^s < d_y^s$, the actual service curve S_y' for flow y required for bounding the delay of flow x is

$$S_y' \geq \frac{\sigma_y}{d_x^s} t$$

The minimum allocated service curves shown in Figure 3.7 required are given by

$$S_x = \frac{\sigma_x}{d_x^s} t$$

$$S_y' = \frac{\sigma_y}{d_x^s} t$$

Let r_y' ($> r_y$) be the higher rate allocated to flow y . The minimum service curve $S_y'(t)$ of flow y becomes $S_y'(t) = r_y' t = \frac{\sigma_y}{d_x^s} t$. The link capacity to support both flows is given by

Theorem 3.3:

$$C_j \geq \frac{\sigma_x + \sigma_y + L_{\max}}{d_x^s} \quad (3.11)$$

Suppose now a request for a new connection z with traffic parameter (σ_z, ρ_z, d_z) arrives. The admission control function has to verify whether condition defined in Theorem 3.3 is true before flow z is admitted. The necessary and sufficient condition for FDPS to provide the service curve guarantees are

$$C_j \geq \frac{\sigma_x + \sigma_y + \sigma_z}{d_i^{s-\min}} \quad \text{s.t. } C_j \geq \rho_x + \rho_y + \rho_z \quad (3.12)$$

where

$$d_j^{s-\min} = \min\{d_k^s\} \quad k \in \{x, y, z\}$$

The end-to-end delay bounds for flows x , y and z are $d_x^f + d_k^{s-\min}$, $d_y^f + d_k^{s-\min}$ and $d_z^f + d_k^{s-\min}$ respectively. We have shown in this example how a new flow could be admitted using Theorem 3.3. The minimum scheduling delay $d_j^{s-\min}$ for all the flows leads to over provisioning to satisfy the lowest scheduling delay constraint of all flows sharing the link.

3.4.4 Difference between RCSP and FDPS

For a better appreciation of the FDPS algorithm, we discuss the subtle differences between RCSP [ZH93] and FDPS in this section. The RCSP algorithm is a static priority scheduling algorithm that applies rate or delay control for each flow by the computation of an eligibility time for each packet of a flow for controlled release of packets. The eligibility time of each packet could be computed based on a delay or rate control criterion.

In other words, an eligibility time is computed for each packet so that it will be released for scheduling once current time is equal or exceeds its eligibility time. Otherwise, the packet is buffered until it is eligible for service. It is a non-work conserving scheduling algorithm. In RCSP, all packets of a flow are assigned the same priority for scheduling. Thus two processes, packet release process and static priority scheduling process, are employed in RCSP.

For the FDPS algorithm, each packet is scheduled according to the state of the flow. By means of a traffic profile in the Service Level Agreement (SLA) of a QoS-based flow, its arrival envelope is deduced to parameterize the ETB. Upon receiving a packet, a dynamic priority, corresponding to state of the flow extracted by the ETB mechanism, is assigned. All packets queued are then scheduled according to their dynamic priority. In other words, bursty packets of a flow would have different priority, a subtle difference from static priority scheduling. The FDPS algorithm is a work-conserving algorithm that reduces burstiness of a flow during the ‘dominant weighted fair queuing’ period of a link discussed earlier. As each packet is marked either as conformant or non-conformant, all conformant traffic is protected against spurious best-effort traffic. Zhang in [ZH93] had also suggested a work conserving RCSP by scheduling non-eligible packets when the output link is idle. Obviously, this work conserving RCSP algorithm is different from the FDPS algorithm due to a static priority assigned to each flow. The FDPS algorithm provides an adaptive scheduling mechanism that protects all conformant traffic equivalent to a dedicated channel and yet allows best effort traffic to utilize spare

capacity in each link. As its performance is bounded by the well-known service curves theory, it provides a per-flow end-to-end QoS guarantees.

We have proved mathematically that the FDPS algorithm realizes Service Curves, we would like to conduct some experiments to verify our results via computer simulation in the next section.

3.4.5 Comparison of Computational Complexities

Computation complexity is one of the key factors that determine whether an algorithm is suitable for implementation. It is a measure of the number of computations needed for a packet scheduling decision in the proposed algorithm. Low computational complexity normally implies simple and low cost implementation. With large number of existing algorithms, the first issue is to identify algorithms that provide deterministic QoS guarantees for comparison with FDPS algorithm. We recall that, in section 3.3, we proved that FDPS provides Service Curves assurance to an admitted flow. Over the past 15 years, Service Curves [BOU01,CR91,CR91+,CR95,CR98], a theoretical approach to deterministic QoS guarantees, was studied extensive and developed. The Service Curves theory establishes the necessary and sufficient conditions for providing per-flow deterministic QoS guarantees. However, to realize service-curve based scheduling is a challenging issue due to difficulties in tracking the real-time arrival and service curves of a flow. The FDPS algorithm is proposed to solve this challenging issue. To the best of our knowledge, there is no practical implementation of service curve scheduling hitherto. The FDPS algorithm is the first successful low-computational-complexity realization of the real-time arrival and service curves of an admitted flow and provides a per-flow deterministic QoS guarantee. Hence, to make compatible complexity comparison, we will compare FDPS with well-known per-flow QoS guarantee algorithms. Only weighted fair queueing (WFQ) [BE96, GO94, GOY97, PAR93, PAR94] algorithms are known to provide per-flow deterministic QoS guarantees.

In WFQ algorithms, a parameter, “the so-called packet finishing time” is computed for every input packet. To determine the order of service in WFQ algorithms, head-of-flow

packets from backlogged flows are sorted in ascending order of the packet-finishing time and the packet with the smallest packet finishing time is scheduled. This operation is repeated for every scheduling decision including packets from any new flow. The computational complexity of such a sorting process in (WFQ) algorithms is $O(\log N)$, where N is the number of backlogged flows. Such computational complexity will become too high for a large value of N .

In contrast to WFQ algorithms, the FDPS implementation employs two dependent processes for packet scheduling: the queueing process and the scheduling process. The queueing process is activated upon the arrival a new packet. The queueing process reads the packet's flow-ETB, computes its flow-state dependent priority and forwards the packet to one of the output queues according the packet's flow-state priority. In the scheduling process, simple priority scheduling selects packets from the backlogged highest priority queue for service. The queueing and scheduling processes have computational complexities of $O(4)$ and $O(1)$ respectively. The combine computational complexity is $O(5)$, much smaller than $O(\log N)$ for WFQ schedulers. The low computational complexity makes FDPS a suitable algorithm for practical implementation.

3.5 Simulation Experiments

In this section, we focus on implementation and evaluation of the FDPS discipline using simulation and implementation experiments based on network simulator NS-2 [NS]. We will demonstrate the advantages and features of the proposed FDPS discipline by examining its various behaviors in a realistic scenario. We will show that the algorithm can provide not only rate but also end-to-end delay guarantee. Through a series of experiments, we give a comprehensive performance evaluation of the proposed scheme.

3.5.1 A simple Simulation Topology

The network topology is shown in Figure 3.8. There are two edge routers and two core router. There are two bottleneck links: one between edge router 1 and core router 1 and another between core router 2 and edge router 2, which has 2.15Mbps bandwidth and

5ms delay. As a result, congestion and thus packet discarding are likely to occur at edge router 1 and core router 2. All other links have 10 Mbps bandwidth and 5 ms delay.

Table 3.1 EXPERIMENT PARAMETERS OF THE TOKEN BUCKET OF THE FDPS SCHEDULER

Parameter	
Token Rate (bps)	1 M
Depth (bit) σ	40000
Queue length	100 packets
Reserved rate ρ	1000Kbps
Average packet length	4000bits

The simulation is for 60 seconds unless otherwise specified. The parameters of the token bucket used in the simulation experiments are given in Table 3.1.

Figure 3.8 Simulation topology 1

Applying Theorem 3.3:

$$C_j \geq \frac{\sum_k \sigma_k + L_j^{\max}}{d_j^{s-\min}} \quad \text{for } \forall j \in \text{Path}_i$$

$$d_i \leq d_i^f + d_i^s$$

where

$$d_i^c = d_i^p + \sum_{\forall j} \frac{L_j^{\max}}{C_j}$$

$$d_i^e \leq d_i^p + d_i^{s-\min} + \sum_{\forall j} \frac{L_j^{\max}}{C_j}$$

where d_i^p is the fixed propagation delay.

For the sake of simplicity, we use a fixed packet size of 2Kbits, thus $L_j^{\max} = 2Kb$. Obviously the propagation delay is $d_i^p = 25ms$. If the minimum scheduling delay is $d_j^{s-\min} = 20ms$, The end-to-end delay is

$$25 + 20 + 2 * 2K / 10M + 2K / 5M + 2 * 2K / 2.15M = 47.8ms.$$

This is the delay bound if only two flows are using the link and have the above parameters.

We first examine the performance of the proposed algorithm when $\sigma - \rho$ traffic is used. In the beginning, this traffic type generates a packet burst which contains σ packets, then followed by a constant rate ρ . Thus, it is conformant traffic and does not need token bucket regulator. $\sigma - \rho$ traffic transmits fixed-size packets.

Case A: Two flows

The σ and ρ of each flow are set to 10 (packets) and 0.9Mb/s. By applying Theorem 3.3, the minimum C_j required by the FDPS algorithm is given by

$$C_j = 2.15Mb/s \geq \frac{2 * 10 * 2K + 2K}{20ms} = 2.1Mb/s \quad \text{the delay bound condition is satisfied. Figure 3.9}$$

shows the delay distribution in this case. Table 2 list the maximum delay for individual flows.

Figure 3.9 Packet delay in FDPS (Case A: two flows, $\sigma - \rho$ traffic)

Case B: Three flows

In the next case, we consider three flows having different parameters. The σ and ρ of flow 1 are set to 10(packets) and 0.9Mb/s, while the σ and ρ of flows 2 and 3 are set to 10(packets) and 0.45Mb/s, respectively. Thus $C_j = 2.15 Mb / s < \frac{3 * 10 * 2K + 2K}{20ms} = 3.1 Mb / s$, the condition is not satisfied. The maximum end-to-end delays are listed in Table 3.2.

TABLE 3.2 MAXIMUM END-TO-END DELAY FOR TWO CASES ($\sigma - \rho$ TRAFFIC)

		Maximum end-to-end packet delay (ms)
Case A	Flow 1	44.437
	Flow 2	45.368
Case B	Flow 1	52.816
	Flow 2	53.746
	Flow 3	54.677

In the above two cases, it is clear that both flows in Case A meet the delay bound but all three flows in Case B could not meet the delay bound. These results verify that the condition according to Theorem 3.3 holds.

On-off traffic:

The experiment is repeated using on-off traffic. The parameters of two flows in Case A are shown in Table 3.3a. For Case B of three flows, the parameters are shown in Table 3.3b. The simulation results on maximum end-to-end delay are shown in Table 3.4. Again, the condition of Theorem is not satisfied in Case B. We further show the delay distribution for the case of three flows when on-off traffic is used in Figure 3.10.

Table 3.3.a EXPERIMENT PARAMETERS OF THE TOKEN BUCKET FOR FDPS
SCHEDULER(CASE A).

Transmission rate	1120 Kb/s
Average On Time	100 ms
Average Off time	40 ms
σ of regulator	10 packets
ρ of regulator	1 Mb/s
Packet Size	2 Kb

Table 3.3.b EXPERIMENT PARAMETERS OF THE TOKEN BUCKET
FOR FDPS SCHEDULER (CASE B)

	Flow 1	Flows 2 and 3
Transmission rate	1120 Kb/s	560 Kb/s
Average On Time	100 ms	100 ms
Average Off time	40 ms	40 ms
σ of regulator	10 packets	10 packets
ρ of regulator	1 Mb/s	500 Kb/s
Packet Size	2 Kb	2 Kb

TABLE 3.4 MAXIMUM END-TO-END DELAY FOR TWO CASES (ON-OFF TRAFFIC)

		Maximum end-to-end packet delay (ms)
Case A	Flow 1	44.036
	Flow 2	41.953
Case B	Flow 1	92.845
	Flow 2	28.612
	Flow 3	29.543

Figure 3.10. Packet delay in FDPS (three flows, On-off traffic)

The results show that the condition in Theorem 3.3 must be satisfied in order to meet the delay bound to be met. In Case B, the delay bound does not hold when the inequality is not satisfied.

3.5.2 A More Complicated Simulation Scenario

Next we conduct the simulation experiments using a network topology shown in Figure 3.11. FDPS is deployed in all links except that the link between core routers 1 and 2 uses the FCFS discipline. Thus this is a mixed configuration in which both FDPS router and conventional FIFO routers are inter-connected as shown in Figure 3.11.

Figure 3.11 Simulation Topology 2

With a fixed packet size of 2Kbits, $L_j^{\max} = 2Kb$, $d_i^p = 25ms$ and $d_j^{s-\min} = 20ms$, the end-to-end delay bound = $25 + 20 + 3 * 2K / 10M + 2K / 3.1M + 2K / 2.15M = 47.2ms$.

A. Simulation results

In this more complicated configuration, we again consider $\sigma - \rho$ traffic in Case A and on-off traffic source shaped by a token bucket regulator in Case B.

With the condition defined in Theorem 3.3 satisfied in Cases A and B, all end-to-end delays satisfy the computed delay bound. We will show that a FIFO core router (router 1) in the configuration does not affect the delay bound provided by FDPS.

There are three flows and the σ and ρ for all three flows are set to 10 (packets) and 0.9Mb/s. The delay distribution for the three flows under $\sigma - \rho$ traffic is shown in Figure 3.12. The maximum end-to-end delay is given in Table 3.6.

Figure 3.12. Packet delay in FDPS (three flows, $\sigma - \rho$ traffic)

B. On-off traffic: Again we repeat the experiment using on-off traffic. Its parameters are listed in table 3.5.

TABLE 3.5 TRAFFIC PARAMETERS FOR ON-OFF TRAFFIC

Transmission rate	1120 Kb/s
Average On Time	100 ms
Average Off time	40 ms
σ of regulator	10 packets
ρ of regulator	1 Mb/s
Packet Size	2 Kb

The packet delay distribution for FDPS under on-off traffic is shown in Figure 3.13. The maximum end-to-end delay is also listed in Table 3.6.

TABLE 3.6 MAXIMUM END-TO-END DELAY FOR TWO CASES

		Maximum end-to-end packet delay (ms)
$\sigma - \rho$ traffic	Flow 1	44.934
	Flow 2	46.214
	Flow 3	33.414
On-off traffic	Flow 1	28.395
	Flow 2	28.251
	Flow 3	21.895

Figure 3.13. Packet delay in FDPS (three flows, on-off traffic)

The maximum delay is quite low, since the burst of flows is “smoothed” by the FDPS scheduler at core router 2. In addition, the bandwidth of links between edge and core router –10Mbps, is much higher than delay required.

3.6 Conclusion

In this chapter, we have presented a new packet scheduling algorithm, namely Flow-state-dependent Dynamic Packet Scheduling (FDPS), that realizes the Service Curves for guaranteed service provisioning. In this scheduling scheme, admitted QoS-based flows are protected yet allowing best effort traffic to co-exist in a multiservices network. The proposed FDPS algorithm performs packet monitoring, marking, scheduling and discarding. With its fine granularity in packet marking, each packet is forwarded and scheduled in a controlled and orderly manner. We have shown through mathematical analysis that the FDPS algorithm defines the arrival and service curves required in

Service Curves and thus allows the theory of Service Curve to be applied in a real network for per flow QoS provisioning. We also gave an associated admission control police example for this guaranteed service. A tight end-to-end delay is also guaranteed for QoS flows in a packet switching network. Compared with existing scheduling algorithms for QoS provisioning, FDPS can provide an integrated guaranteed and best effort service in a multi-services network. We have conducted extensive simulation experiments based on NS-2 and the numerical results validate our theoretical analysis.

CHAPTER 4

Congestion Control Based on Flow-state-dependent Dynamic Priority Scheduling

The quest for reliable data transmission in packet switched networks has spurred much research interest in TCP-friendly transmission control. The existing congestion control mechanism in current Internet is based on the TCP congestion control mechanism which is the key to the success of the Internet. TCP-friendly environment assumes all flows to be TCP-friendly - a source increases its rate if bandwidth is available and decreases its rate when congestion arises. However, the TCP's end-to-end congestion control mechanism and the router assistant mechanism Active Queue Management (AQM), such as Random Early Detection (RED), cannot cope with flows that are unresponsive to congestion notification or non-TCP-compatible flows which are responsive but more aggressive than TCP. Unresponsive or non-TCP compatible flows could even cause congestion collapse. As TCP-friendly environment is a source dependent implementation, it is not clear whether congestion is best handled at the transport layer. In this chapter, we present a new mechanism, namely Source Aware Congestion Control (SACC), in which congestion is handled at the packet scheduling layer. The SACC mechanism is based on Flow-state-dependent Dynamic Priority Scheduling (FDPS), which performs traffic monitoring, marking and scheduling at routers. The proposed SACC mechanism has the following properties: 1) conformant traffic is guaranteed while non-conformant traffic is discarded gracefully when congestion arises; 2) it provides protection against source unresponsiveness; and 3) TCP-compatible protocols and AQM, such as RED may inter-work with this mechanism without modification. We provide analysis of the congestion handling mechanism and simulation results to show the behaviors of SACC in a variety of scenarios. We have conducted extensive simulation experiments with real video trace and TCP flows based on NS-2 to examine the SACC behaviors and evaluate its performance. The numerical results illustrate that the proposed scheme can differentiate

services and resolve a congestion by gracefully discarding packets from non-conformant and best effort flows.

4.1 Introduction

As the Internet grows in leaps and bounds with rapid increase in the number of users and applications, the concentration of traffic at some nodes of the network causing congestion becomes a main concern. The current Internet has suffered from the problem of congestion which is inherent in best effort datagram networks due to uncoordinated resource sharing. Traditionally, in the Internet all packets are treated the same without any discrimination or explicit delivery guarantees. This is known as the *best effort service model*; the network promises to exert its best effort to deliver the packets injected into it without committing to any quantitative performance (QoS) bounds. There is no coordinated resource sharing mechanism in the best-effort service model. Therefore, the perceived performance is determined not only by the network itself, but also from other users' offered load, resulting in a complete lack of isolation and protection. Though the traditional applications and protocols were flexible, adaptive, and robust enough to operate under a wide range of network conditions without requiring any particularly well-defined service, the problem of congestion has become more and more serious, along with the introduction of more applications to the Internet, such as emerging streaming traffic.

The Internet protocol architecture is based on a connectionless end-to-end packet service using the IP protocol. In order to deal with congestion, the Internet uses end-to-end window-based flow control in its Transmission Control Protocol (TCP) [JA88, JA90, CHI89, LO02], primarily for controlling demand on the receiver's bottleneck resources. TCP as the main transport protocol used in the Internet employs congestion avoidance and congestion control mechanisms to deal with the congestion problem. In congestion avoidance [JA88,JA90], the idea is to monitor the state of the network and generate feedback to all sources sharing the bottleneck link to increase/decrease their rate according to the change of network states [JA88]. The TCP congestion control

mechanisms have served the Internet quite well for traditional applications and formed the basis for its survival and success.

However, recent developments in the diversity of application requirements make it obvious that a diverse range of services delivered to the users is needed. The traffic flows in current Internet can be roughly classified into three classes: (1) TCP-compatible flows, (2) unresponsive flows, i.e., flows that do not slow down when congestion occurs, such as UDP flows and (3) flows that are responsive but are not TCP-compatible. The AQM or ECN-enabled AQM [FL93, IETF98, IETF01] can be used to benefit the TCP-compatible flows. However, TCP's end-to-end congestion control mechanism and the router algorithm AQM, such as RED, cannot deal with flows that are unresponsive to congestion notification or non-TCP-compatible flows which are responsive but more aggressive than TCP. The non-responsive and non-TCP-compatible traffic contains more aggressive flows that pose significant threats to the Internet performance, as an unresponsive flow in this network could cause a new congestion collapse.

The legacy Internet architecture assumes that all flow-related state should be kept at the hosts; therefore, the congestion control mechanisms hitherto have been mainly implemented at the end hosts. However, with decisions made at the end hosts and treatment of the network as a black box that simply drops packets, there is clearly a limit on control over the allocation of network resources. This also limits the range of services the network is capable of offering. Routers, on the other hand, know exactly how congested links are and can therefore perform more intelligent resource management. Thus, the introduction of router mechanisms for congestion control enables the network to manage its own resources more efficiently, which is a reasonable choice.

A previously proposed way to use network (router) support for congestion control is to change the scheduling discipline inside the routers. For example, per-flow packet scheduling algorithm, such as Fair Queuing scheduling [PAR93] is employed to provide fair allocation of bandwidth to competing flows. However, the research community does not agree on the utility of this scheduling discipline for congestion control, even if its

flow isolation property is appreciated, nor on the way to use this scheduling discipline [AR02]. The reasons include its high implementation complexity and the uncertainty on how WFQ can be used to handle best-effort and QoS traffic concurrently existing in the network. Other examples include core-stateless fair queuing (CSFQ)[STO98], CHOKE [PA00], etc., which try to approximate the WFQ. However, these mechanisms could reduce the likelihood of congestion and unfairness in the Internet, but do not eliminate them. Moreover, these mechanism are much more complex to implement than the simple FIFO queuing.

In the past years, some researchers have studied the problems of congestion and unfairness caused by unresponsive flows such as streaming media [HO01]. Recently, C. Albuquerque, et.al., proposed a congestion-avoidance mechanism called network border patrol (NBP) [AL04], which aims at dealing with unresponsive flows to network congestion. The main idea of NBP entails the exchange of feedback between routers at the network borders to detect and prevent unresponsive flows from entering the network. However, this method needs the edge routers to perform per-flow control and feedback exchange operation. It also does not support both guaranteed and non-guaranteed flows co-existing in the network.

In this chapter, we try to provide a new solution to the congestion control problem using packet scheduling method. It is a router-assistant scheme, which is implemented in the packet scheduling and buffer management domain. In our proposed congestion control mechanism, each router in the network implements a simple Flow-state-dependent Dynamic Priority Scheduling (FDPS) Algorithm. In this network, the users may act selfishly as they try to maximize their own satisfaction without taking into account other users. That is, the proposed SACC scheme can cope with flows that are unresponsive to congestion notification or non-TCP-compatible flows, which are responsive but more aggressive than TCP. It uses a mechanism that identifies each non-TCP-compatible flow and discards its packet and forces it to reduce its rate.

This new scheme relies neither on the end-to-end congestion control mechanism nor intends to replace or modify TCP. Instead, it is designed to be compatible with TCP and

does not violate the end-to-end argument due to the network support required. It can work in conjunction with an end-to-end congestion control mechanism to cope with the non-responsive traffic flows by discarding the “right” packets of non-conformant and best effort traffic during congestion. The proposed scheme can also work in conjunction with the current TCP congestion avoidance mechanism and active queue management mechanism, as a component of the future QoS provisioning architecture in the Internet. SACC can differentiate services for different traffic flows and resolve the congestion by gracefully discarding packets from non-conformant and best effort flows. Another interesting feature of the SACC mechanism is that it approximates WFQ behaviors if there are only best effort flows. In this case, SACC approximates fair max-min share behaviors to provide fairness among best-effort traffic flows. We will show that SACC exhibits very good features through analysis and simulation experiments.

The rest of this chapter is organized as follows. Section 4.2 introduces the concept of source-aware congestion control and gives an overview of the proposed SACC scheme. We review the SACC mechanism based on a Flow-state-dependent Dynamic Priority Scheduling(FDPS) algorithm with examples to illustrate its operation in Section 4.3. Section 4.4 presents in detail the implementations of the proposed SACC and its performance evaluation. In Section 4.5, we examine the behaviors of SACC in the case that only best-effort traffic flows exist in the network. Finally, we conclude this chapter in Section 4.6.

4.2 Overview of the Proposed Source Aware Congestion Control (SACC)

In contrast to the existing congestion avoidance/control methods [JA88, JA90] based on end-to-end mechanisms or by using AQM, the proposed Source Aware Congestion Control (SACC) mechanism is implemented at the scheduling and buffering management layer. In this mechanism, a source is made aware of its guaranteed reserved rate but is

allowed to transmit non-conformant traffic; when there is excess network capacity available along the path of its connection, all conformant and non-conformant traffic is delivered to their destination. Once congestion occurs, non-conformant traffic is discarded gracefully where the worst non-conformant traffic (defined as the most negative in priority in SACC, which will be elaborated in Section 4.4) will be discarded with the highest precedence.

Such a source-aware congestion control mechanism allows a source to transmit more traffic than its reserved rate of its SLA knowing that the delivery of the conformant traffic is guaranteed. A useful application involves video coding for MPEG files. The total codes may consist of a base layer code transmitted as conformant traffic while multi-enhancement layer codes are transmitted as non-conformant traffic. When the network is lightly loaded, all packets are delivered to their destinations. However, if congestion arises along the path of the flow, the base layer codes will be guaranteed while the enhancement layers codes will be delivered whenever unused link capacity becomes available, starting from the first enhancement layer.

As the mechanism identifies conformant and non-conformant traffic, it facilitates efficient network utilization at low guaranteed load by carrying more non-conformant and best effort traffic but protects all conformant traffic when congestion arises. With the SACC mechanism, the delivery of conformant traffic is guaranteed while non-conformant and best effort traffic is delivered via best effort service. The formal definitions of conformant and non-conformant traffic have been given in Chapter 3. With the definitions, it is clear to see that a QoS-based application may introduce both conformant traffic and non-conformant traffic to the network. The non-conformant part, which is identified and marked by the proposed SACC mechanism, will be treated as best-effort at the router by the proposed scheme. In other words, the best-effort traffic in the discussions for the rest of the paper may include the traffic from best-effort services and the non-conformant part of the traffic introduced by QoS-based applications.

In the SACC mechanism, an enhanced leaky-bucket is employed to generate a dynamic priority for each packet. Each conformant packet is assigned a positive dynamic priority while each non-conformant packet is assigned a negative dynamic priority. The positive

and negative priorities provide fine granularity in the demarcation of conformant and non-conformant packets. A packet scheduling algorithm, called Flow-state-dependent Dynamic Priority Scheduling (FDPS) algorithm proposed in Chapter 3 is employed as the key component in the SACC scheme. The FDPS assigns the dynamic priority and performs the packet scheduling, buffer management and packet discarding.

The notion of marking packets according to their traffic profile using a coarse granularity is not new. However, the FDPS algorithm marks packets with fine granularity for discarding the packets with negative dynamic priority when congestion arises. Packets are queued according to their dynamic priority where the one with the lowest priority is queued at the end. We will show that “Drop tail” scheme is able to remove the most non-conformant packets at every moment of a new packet arrival.

Although there are many congestion control schemes ranging from end-to-end mechanisms to router assistant mechanisms proposed in the literature, there is little known work that integrates both guaranteed and non-guaranteed traffic in a network in the context of congestion control. For a network carrying integrated guaranteed and non-guaranteed traffic, protection of guaranteed traffic is crucial for the usefulness of any scheme. The proposed FDPS algorithm caters to such an environment. In the case that there is residual bandwidth available for non-guaranteed flows, the FDPS algorithm allocates the available bandwidth in an approximate Fair Queuing manner. The SACC mechanism starts to discard packets from the queue with lowest dynamic priority when the queue size exceeds a predefined threshold. One of the most useful features of the proposed SACC mechanism is its capability to inter-work with current TCP congestion control mechanism without modification. We will use extensive NS-2 simulations and analysis to illustrate the operation of the SACC mechanism.

Let us use an example to illustrate how the SACC mechanism works. Consider a flow transmitting both conformant and non-conformant traffic. In order to ensure that the “conformant part” of the traffic is conformant, its traffic is policed (or monitored) by a regulator in the source while each non-conformant packet is marked with a “non-

conformant” bit. One implementation example is to use one bit of the “Type of Service” field on the IPv4 header or the “Traffic Class” field in IPv6 header for this purpose. By means of the proposed FDPS algorithm, conformant packets are assigned positive dynamic priority while non-conformant packets are assigned negative dynamic priority. Both conformant and non-conformant packets could then be interleaved in the flow and sent to their ingress node. The conformant traffic is guaranteed by the FDPS algorithm that provides isolation between conformant and non-conformant traffic using the positive and negative dynamic priority. Each source is aware of the guarantee provided by the network and its non-conformant traffic is discarded when congestion arises. A user is therefore able to enter a service contract most appropriately for his requirements yet having the flexibility of sending non-conformant traffic to exploit the excess bandwidth along the path of its connection.

4.3 Mechanism and Algorithm of Source Aware Congestion Control (SACC) Based on FDPS

In this section, we first present the FDPS algorithm and show how this algorithm may be used by the SACC mechanism. We recall that the SACC mechanism is a congestion control mechanism in which both TCP-compatible and non-TCP-compatible (e.g. UDP or other flows) may co-exist in the network. The SACC mechanism must provide the protection for guaranteed traffic and yet provide fair sharing of residual link capacity for non-guaranteed traffic independent of whether it is TCP-compatible traffic or not. We will show this is provided in SACC.

The FDPS algorithm is a traffic monitoring, priority marking and scheduling algorithm. Packets from each flow is monitored by the algorithm for conformance status and marked accordingly. For each conformant packet a positive dynamic priority is assigned to the packet. The marking of the packet facilitates them to be forwarded to the corresponding dynamic priority queue at the output buffer. A total of p^+ output queues are available to conformant traffic and p^- output queues for non-conformant traffic. Note that p^+ and

p^- also indicate the maximum and minimum dynamic priority assigned to a packet. The details of the operations of FDPS have been given in Chapter 3.

The SACC mechanism utilizes packet discard to control congestion at any output link j . As non-conformant packets are queued behind conformant ones in the output queues of Figure 3.2, link congestion can be alleviated by setting the appropriate discard threshold Th_j for the output buffers. A virtual buffer may be constructed as the concatenation of all the output queues for a link, i.e, $p^+ + p^-$ queues. The total queue length in this virtual buffer is the sum of all the packets in all the real queues. In this mechanism, packets are discarded on-the-fly when congestion arises. Hence the correct queuing sequence is a necessary condition for drop-tail to work well. In the traditional “drop-tail” scheme, new packet arriving at a full queue may be discarded independent of its traffic profile. In the FDPS algorithm, a queue is maintained using a virtual buffer using all the queues in Figure 3.2. If a packet arrival causes a packet discard threshold Th_j^d of link j to be exceeded, the packet at the end of the lowest priority queue is discarded. The conformant traffic of each flow is regulated by a leaky-bucket (σ_i, ρ_i) at its source. We will derive the minimum buffer requirement B_j^m needed for any output link j to buffer all conformant packets.

Lemma 4.1 For a new request for flow i , admission control function admits a flow if and only if the service curve $S_i(t)$ is guaranteed at every link along its path $PATH_i$

$$\sigma_i + \rho_i t \leq S_i(t + d_i) \quad (4.1)$$

The above condition ensures that the minimum level of guaranteed service along the path of flow i . If this service curve is guaranteed at each link, the service delay of every packet is at most d_i at a link.

Lemma 4.2 For every link j , the necessary condition for all flows using link j is

$$\sum_{i \in F_j} S_i(t) \leq C_j(t) \quad (4.2)$$

F_j is the set of all guaranteed flows sharing link j .

The service curve for the aggregated traffic using link j is given by $\sum_{i \in F_j} S_i(t)$. Hence the capacity of the link must be sufficient to provide the total amount of service. In other words, the maximum service curve $C_j(t)$ of link j is the envelope of the sum of all service curves $S_i(t)$ in the worst case.

Lemma 4.3 Guaranteed flow i is admitted only if the following condition is true at each link along its path $PATH_i$

$$\sum_{i \in F_j} (\sigma_i + \rho_i t) \leq \min\{C_j(t + d_i)\} \quad \forall j \in PATH_i \quad (4.3)$$

where d_i is the maximum delay allowable for flow i at link j , F_j is the set of all guaranteed flows sharing link j , C_j is the capacity of link j , and $PATH_i$ is the set of links in the path of flow i .

The purpose of inequality (4.3) is to ensure that the arrival traffic envelope, i.e., $\sum_{i \in F_j} (\sigma_i + \rho_i t)$ when right shifted by interval $\min_{i \in F_j} \{d_i\}$ is below the maximum service curve, $C_j t$, of the link j .

From inequality (4.3), we have

$$\frac{\sum_{i \in F_j} \sigma_i}{C_j} \leq \min\{d_i\} \quad \text{for } \forall j \in P_i$$

As the focus of this chapter is on congestion control, we will not discuss the issue of delay guarantees due to Lemma 4.3 any further but assume that the above condition is satisfied if a flow is admitted.

Lemma 4.4 The minimum buffer size B_j^m to prevent any loss of conformant packets at the output buffer of link j is given by

$$B_j^m = \sum_{i \in F_j} \sigma_i \left(1 - \frac{C_j - \sum_{i \in F_j} \rho_i}{C^* - \sum_{i \in F_j} \rho_i} \right) \quad (4.4)$$

where F_j is the set of all flows using link j , C^* is the aggregate rate of all input links to the node, C_j is the capacity of link j .

Proof: Note that the maximum arrival rate to the output buffer of link j is C^* . The leaky-bucket and practical traffic envelopes and service envelope of link j are shown in Figure 4.1.

Let t_Δ be the maximum burst duration for which packet arrives at rate C^* . From Figure 4.1, we can see that

$$\sum_{i \in F_j} \sigma_i + \sum_{i \in F_j} \rho_i t_\Delta = C^* t_\Delta \quad \text{or} \quad t_\Delta = \frac{\sum_{i \in F_j} \sigma_i}{C^* - \sum_{i \in F_j} \rho_i}$$

The minimum buffer size B_j^m should allow a buffer to store all packet backlogged after interval t_Δ as this is the interval with maximum amount of backlogged packets accumulated. We have

$$\begin{aligned} B_j^m &= C^* \frac{\sum_{i \in F_j} \sigma_i}{C^* - \sum_{i \in F_j} \rho_i} - C_j \frac{\sum_{i \in F_j} \sigma_i}{C^* - \sum_{i \in F_j} \rho_i} \quad \text{s.t. } \sum_i \rho_i \leq C_j \\ &= \sum_{i \in F_j} \sigma_i \left\{ \frac{C^* - C_j}{C^* - \sum_{i \in F_j} \rho_i} \right\} \end{aligned}$$

Rearranging it yields

$$B_j^m = \sum_{i \in F_j} \sigma_i \left(1 - \frac{C_j - \sum_{i \in F_j} \rho_i}{C^* - \sum_{i \in F_j} \rho_i} \right) \quad \text{s.t. } \sum_i \rho_i \leq C_j$$

and this completes the proof.

Lemma 4.4 is useful in determining the required minimum buffer allocation to a link subject to the input link capacity. Note that if $C^* = C_j$, no buffering is needed as the input

rate equals the output rate. If $C_j \rightarrow \infty$, $B_j^m \rightarrow \sum_{i \in F_j} \sigma_i$ is the theoretical buffer requirement for output link i with large number of input links.

Thus, all conformant packets marked by the FDPS algorithm are guaranteed with sufficient buffer space even in congestion. However, non-conformant packets queued behind these conformant packets may be discarded when congestion arises. Thus the minimum buffer size B_j^m provides protection against any loss of guaranteed traffic.

Figure 4.1 Theoretical and practical traffic envelopes

Theorem 4.1 The discard threshold Th_j^d for the output buffer of link j that protects against any loss of conformant packets using the SACC mechanism is given by

$$Th_j^d \geq B_j^m$$

This result follows directly from Lemma 4.4. The smallest threshold is given by $Th_j^d = B_j^m$. This smallest discard threshold $Th_{j(\min)}^d$ is used to discard any packet queued behind this virtual queue length as this packet is obviously a non-conformant packet and should be discarded during congestion.

Hitherto, we have discussed the buffer requirement for the conformant traffic. The remaining issue is the requirement for the non-conformant traffic. The purpose of the

buffering is to facilitate best effort traffic to co-exist with the QoS-based flows in a multi-service packet switching network and to discard such traffic when congestion arises. If the link is not congested, the total queue length could be very small. On the other hand, if all conformant flows burst together, the maximum queue length is given by $Th_{j(\min)}^d$. Non-conformant packets would be discarded if a congestion persists. The following procedure is applied for discarding non conformant packets:

Q_j^{tot} is the total queue length (including positive priority queues and negative priority queues. Q_j^{be} is the length for the negative priority queues (for non-conformant traffic). Th_j^{be} is the packet discarding threshold for the non-conformant traffic.

If $Q_j^{tot} \geq Th_{j(\min)}^d$, discard the most non-conformant packet until condition becomes false

If $Q_j^{be} > Th_j^{be}$ and $Q_j^{tot} < Th_{j(\min)}^d$, discard the most non-conformant packets until condition becomes false

The first condition satisfies the condition for guaranteed flows stated in Lemma 4.4. In the Lemma, the buffer threshold ensures that no conformant packet is discarded even if guaranteed flows at a link start bursty transmission together. The second condition deals with the case in which packet accumulation is due to aggressive best-effort traffic. In such a case, we will discard packets from best-effort traffic even if the first condition is false. In other words, the two conditions facilitate packet discarding due to 1) congestion due to simultaneous bursty transmissions of guaranteed traffic and 2) congestion due to aggressive best-effort traffic. This procedure allows both conformant and best-effort traffic to share the link and best-effort traffic to be discarded whenever congestion arises.

A small queue length for best effort traffic would be sufficient to cater to the transient condition. From our experiments, we observe that $Th_j^{be} = 20$ is sufficient.

4.4 Implementation and Evaluation of the SACC Scheme

In this section, we focus on evaluating the proposed SACC scheme using simulation and implementation experiments based on network simulator NS-2 [NS]. We will demonstrate the advantages and features of SACC by examining its various behaviors in a realistic scenario where traffic flows of VBR videos and TCP traffic co-exist in a network. The video streams carried by UDP flows are non-responsive while the TCP flows are responsive in which the source will adjust the transmission rate when a packet loss is detected. Through a series of experiments, we give a comprehensive performance evaluation of the proposed scheme.

Figure 4.2 Simulation network Topology

4.4.1 Simulation Settings

The network topology is shown in Figure 4.2. There are two edge routers and one core router. The bottleneck link is the one between the core router to the edge router 2 which has 3Mbps bandwidth and 5ms delay. As a result, congestion and thus packet discarding

are likely to occur at the core router. All other links have 10 Mbps bandwidth and 5 ms delay. Two video flows and two TCP flows of FTP are carried (New Reno is adopted as the TCP protocol). We use two real VBR video traces to generate the non-responsive traffic in these simulations, each having an average data rate of 500Kbps. The TCP connections are used for a large file transmission via FTP, each with a maximum window roughly equal to the delay-bandwidth product, which is 40 packets.

Figure 4.3 shows instantaneous and average queue length at the bottleneck router where the congestion occurs when legacy drop tail buffer management and the SACC schemes are used respectively. We can observe from this experiment that the fluctuation (varying range) of the queue length for drop tail is much larger than that for SACC: from 0 to about 100 packets for the former while from 0 to about 20 packets for the latter. The average queue length for SACC is significantly smaller than that for drop tail. These behaviors illustrate that the SACC can provide better congestion control for the bottleneck router for bursty traffic and provide much smaller overall average packet delay for the flows. The SACC mechanism identifies the most aggressive flow and drops its packets to trigger rate adjustment at their sources using TCP congestion control avoidance mechanism. Other parameters of SACC are listed in Table 4.1.

TABLE 4.1 EXPERIMENT PARAMETERS
OF THE TOKEN BUCKET OF THE SACC SCHEDULER

Parameter	Video	TCP
Token Rate (bps)	1 M	700 K
Depth (bit) σ	40000	8000
Queue length	100 packets	100 packets
Reserved rate ρ	1000Kbps	500Kbps
Average packet length	4000bits	8000bits

We simulate for 60 seconds of time if not specified. The parameters of the token bucket used in the SACC in the simulation experiments are given in Table 4.1. We simulate a

series of scenarios including the video sources are regulated; RED gateway algorithm is applied to the TCP flows on top of SACC, etc. In the following, we give detailed discussions of different scenarios and present the results.

4.4.2 Experiment 1: Non-Regulated VBR Video and TCP Connections

In the first experiment, we compare the behaviors and performance of the SACC with those of the legacy routers adopting drop tails as the buffer management policy. The video sources are regulated with an average rate of about 500Kbps. The video traffic is carried by UDP flows which are non-responsive. That is, there is no end-to-end flow control mechanism. Due to the fluctuations of the VBR traffic, a small part of bursty video traffic will be identified as non-conformant traffic by the SACC scheduler and could be discarded during congestion at the core router.

Figure 4.3 Bottleneck Queue Length (VBR video + TCP flows)

Figure 4.4 compares the packet loss ratio for the video flows when drop-tail, weight-round-robin (WRR) and SACC are employed respectively. Figure 4.4 shows that the video packet-loss ratio of WRR falls between those of drop-tail and SACC. As the video source is VBR and there is no traffic shaper to regulate the traffic, some of the bursty traffic of the VBR video will be identified as non-conformant by the SACC scheduler. We use this video to emulate a traffic source with both conformant and non-conformant traffic. During 10ms and 20ms, there is a burst in the video source and severe congestion occurs, and thus many packets will be discarded. We can observe that the packet loss ratio for drop tail is almost always higher than those of WRR and SACC, especially during the congestion period. Actually, the drop tail policy treats all the packets of both TCP and video traffic in the same manner. It does not provide any traffic differentiation. When a congestion happens, the video and TCP flows compete for the bandwidth resources. During the congestion period when a burst of video comes, more packets of video will be discarded. For WRR, the bandwidth is shared proportionally among video and TCP. It does not provide differentiation between conformant and non-conformant traffic. In contrast, the SACC mechanism only sacrifices the TCP traffic while guaranteeing the conformant traffic. In more detail, the SACC discards TCP packets as best effort service and TCP sender will backoff according to the end-to-end congestion control mechanism. As a result, the video can use up any unallocated bandwidth and thus the packet loss ratio is significantly smaller than that for drop tail scheme.

Figure 4.4 Packet loss ratio for drop tail and SACC scheme (VBR video+TCP)

Figure 4.5. Packet delay statistics for video and TCP connections using drop tails and SACC schemes

Figure 4.5 illustrates the Cumulative Distribution Function of the packet delay. We can see that SACC can provide best delay performance for video traffic, among the three schemes, at the expense of the TCP delay performance being the worst among the schemes. This compromise is reasonable as the TCP traffic as the best effort service has lower priority. Moreover, TCP traffic is usually used to support non-time-sensitive applications, such as FTP, HTTP, etc., and this tradeoff is thus reasonable. The fine granularity in packet marking mechanism in SACC causes the best effort traffic to be serviced last or discarded when congestion arises. The graphs in Figures 4.4 and 4.5 show that SACC mechanism is very responsive to changes in guaranteed traffic.

4.4.3 Experiment 2: Regulated VBR video and TCP connections

In Section 4.3, we have mentioned that SACC has a very attractive feature: when there is congestion, SACC will guarantee the transmission of conformant traffic by gracefully discarding the packets of non-conformant traffic. In this experiment, we add a token bucket as the traffic shaper at each video source. The token bucket has token rate of 1Mbps and depth of 10 packets. These parameters imply that the peak rate of VBR video flows is 1Mbps and thus all video traffic is conformant. The scheduler of SACC should guarantee the video traffic.

Figure 4.6 Packet loss ratio for drop tail and SACC scheme (regulated VBR video+TCP)

Figure 4.6 depicts the packet loss ratio for video traffic when a token bucket shaper is used and when it is not used at the source in the SACC scheme respectively. It is clear that the loss ratio is zero when a token bucket is used while the ratio fluctuates above 0 when there is no token bucket. This result is as expected as all the regulated video traffic is conformant. This experiment demonstrates an important feature of SACC: it can guarantee the transmission of conformant traffic. During congestion, non-conformant traffic is sacrificed and the reserved rate for conformant traffic is maintained.

Figure 4.7 Packet delay statistics for video and TCP connections using drop tails and SACC schemes

In Figure 4.7, we show the delay statistics of the video and TCP traffic when the videos are regulated and when they are not regulated using a token bucket respectively. We can see that the delay performance for video is very close for the two cases while the delay performance for TCP when there is no shaper is better than that when there is a shaper.

4.4.4 Experiment 3: Non-Regulated VBR Video and TCP Connections with RED Gateways

The third experiment is to examine the behaviors of SACC working in conjunction with RED gateways. In this simulation scenario, the RED algorithm is applied to the logical queue of TCP packets which belong to the responsive best effort traffic. The gateway parameters are set as follows: minimum threshold $min_{th} = 15$ packets, maximum threshold $max_{th} = 50$, maximum probability $max_p = 0.02$, and queue weight $w_q = 0.002$.

The results shown in Figure 4.8 illustrate that when there is RED gateway for TCP connections, the average queue length is always greater than that using SACC only. The difference is not very large. The influence of RED gateway on SACC is thus not very significant.

Figure 4.8. Comparison of bottleneck Queue Length with and without RED gateways

Figure 4.9 shows the comparison of packet loss ratio of video traffic with and without RED gateway. We can see that the packet loss ratio with RED gateway is significantly smaller than that without RED gateway during the time interval 0 to 18s. The difference is minor after 18s. This is because FDPS will start to drop packets after the queue length is greater than $\sigma_1 + \sigma_2 = 2\sigma$ while RED gateway starts to drop packets only when the average queue length is larger than the minimum threshold. In other words, the queue length in RED gateway is greater than that in SACC during 0 to 18s. If each of the nodes uses a minimum buffer size as defined in Theorem 4.1 in Section 4.3, some non-conformant packets will be discarded when congestion arises at a link.

Figure 4.9. Comparison of packet loss ratio of video traffic with and without RED gateway

Figure 4.10 shows the comparison of packet delay statistics with and without RED gateway. An interesting observation is that the delay statistics for both TCP and video are very close with a RED and without a RED gateway algorithm. The difference is almost invisible, except when the delay is larger than 3s, SACC TCP has better performance than that for SACC with RED TCP. This illustrates that RED has no significant influence on the delay performance when it is used in conjunction with the SACC scheme. As SACC could identify the right packets for discarding during a congestion, there is little effect of having RED running on top of SACC scheme for a congested network.

Fig.10 Comparison of packet delay statistics with and without RED gateway

4.5 The Behaviors of SACC in Best-Effort Traffic Environment

Although the FDPS algorithm can be used in an integrated guaranteed and best effort service environment, it would be interesting to examine the behaviors of SACC in a packet switching network which provides only best effort service, a typical environment in the current Internet.

Without admission control in a best effort environment, every new connection request is accepted. A typical scenario is a TCP/IP network carrying both TCP and UDP traffic. In this case, we will show that the SACC mechanism can provide approximately the fair sharing of available bandwidth, and could handle a congestion at a link by discarding the right packets, the most non-conformant ones. In addition, a bursty flow could grab additional capacity only when large unused bandwidth is available.

Let us use an example to analyze the behaviors of SACC in this environment. We consider a case in which three best effort flows are sharing a bottleneck link. Let the three flows be denoted by f_1 , f_2 and f_3 respectively. In this best effort environment, a mean rate, denoted by r_m is defined. This mean rate r_m controls the token generation rate for each flow. We also assume that all 3 flows use the same mean packet size l but may have different rates, say r_1, r_2 and r_3 , respectively without any loss of generality. We further assume that $r_1 \leq r_2 \leq r_3$ where r_1 is the lowest flow rate in this environment.

We first consider the case in which $r_1 = r_2 = r_3 \leq r_m$ with no congestion. If the token generation rate r_m is greater than the packet arrival rate of a flow, excess tokens are discarded when the token bucket is full. As all packets will be given a dynamic priority of -1, all the packets from these 3 flows f_1 , f_2 and f_3 are forwarded to the queue with $p = -1$. This is equivalent to a first-come-first-serve (FCFS) service as all packets are sent to this output queue in the order of their arrivals.

We investigate the next case in which $r_1 \leq r_m \leq r_2 \leq r_3$. In this scenario, we intuitively deduce that the packets of flow f_1 will be forwarded to queue $p = -1$, whereas packets of flows f_2 and f_3 would go to queue $p = -1$ and those queues with $p \leq -2$. Hence, when a congestion arises, there is a clear differentiation between flows f_1 , f_2 and f_3 . Packets from f_2 and f_3 at the lower priority queues are easily identified for preferential discard. In our SACC implementation, the last packet of the lowest priority queue will be discarded first. The TCP congestion control procedure would then be triggered.

Now let us examine a congestion scenario in which flows f_2 and f_3 become very bursty. In this situation, all packets will be served when excess capacity is available. However, if f_2 and f_3 are very aggressive for a prolong period and the sum of all flow rates is greater than the link capacity, the buffer will eventually overflow. Again, we can analyze intuitively where the packets are buffered. As f_1 is transmitting at rate $r_1 \leq r_m$, its packets

are queued at queue $p = -1$. Whereas packets from flows f_2 and f_3 go to queues $p = -1$ and $p = -2, -3$ and so on. Suppose there are n queues available, then output queue $p = -n$ would be used. All packets in queue $p = -1$ are packets from flows transmitting at or below mean rate r_m and packets in queues of lower priority are due to flows that are transmitting at a higher rate. Hence, when a congestion arises, packets in the lowest priority queue would be discarded to trigger TCP congestion control, allowing flows to readjust their transmission rates.

The mean rate r_m and the number of negative priority queues for the best effort traffic are yet to be determined. We argue this is not a major issue as r_m can be set to the lowest rate expected in the TCP environment, say 50kbps. It allows the identification of packets from flows which are too aggressive for discard. The number of negative priority queues, n is also the depth of the negative token bucket. If this depth n is large, a flow silent for a long time may hog the link for $n-1$ packets in relation to other aggressive flows. On the other hand, too small a value of n leads to coarse granularity for identifying the aggressiveness of each flow. Actually, in the extreme case where $n=1$, the SACC reduces back to the “drop-tail” case where all packets from different flows share one queue. An approximate reasonable choice of n allows a source to differentiate between flows with different aggressiveness. According to our experiment, the value of n between 5 and 10 is an appropriate choice.

We use NS-2 simulations to verify the above analysis. The network topology we use is identical to that in Section 4.4. But now all flows are best-effort flows. In the first experiment, three TCP flows, with RTTs of 40ms, 140ms and 280ms respectively, compete for the bottleneck link bandwidth of 3Mbps. Other parameters of the SACC scheme are the same as those in Table 4.1. In this case, it is known that the TCP flow with a shorter RTT will obtain a bigger share of the bottleneck bandwidth, if a FIFO with Tail-drop algorithm is used in the router. Figure 4.11 shows the evolutions of the three TCP flows. As illustrated the TCP flow with a shorter RTT has an advantage over those with longer RTTs. As a comparison, Figure 4.12 shows the evolutions of the TCP flows

when SACC scheme is applied. We observe that the three curves are very close, which indicates that the three flows obtain approximately the equal bandwidth shares at the bottleneck link. This is because the SACC using FDPS forces the more “aggressive” flows to backoff more and thus eliminates the influence of RTT on the bandwidth sharing. The more aggressive flows with shorter RTT experience more packet discards and thus backoff more, leading to fair sharing of bandwidth. This is a very interesting feature of SACC as it can eliminate the influences of RTTs on the fairness in TCP. This behavior is similar to that of the Fair Queuing.

Figure 4.11 Bandwidth shares of three TCP flows using FIFO with tail-drop router
(Three TCP flows)

Figure 4.12 Bandwidth shares of three TCP flows using FDPS in the router
(Three TCP flows)

In the second experiment, we consider one UDP flow (CBR) with rate 2.5Mbps and two TCP flows, with RTTs of 40ms and 140ms respectively, competing for the 3Mbps bottleneck link bandwidth. Figure 4.13 shows the bandwidth sharing among the three flows when FIFO with tail-drop is applied. We can clearly see that the non-responsive UDP flow almost monopolizes the bottleneck bandwidth and the TCP flows obtain very little share as a result when FIFO with tail drop is applied. Figure 4.14 shows the bandwidth shares of the three flows when SACC with FDPS is applied (where the Y-axis represents the “number of packets received by the receiver” for UDP flow). We can observe that the two TCP flows obtain much more shares compared to the case when FIFO with tail-drop applies. The non-responsive UDP flow has been penalized by the SACC scheme by dropping some UDP packets. But the UDP flows still obtain more share than TCP flows. This difference of the bandwidth share is due to the inherent fluctuation behaviors of TCP flows. That is, the TCP flows cannot make full use of its share of bandwidth. In this case, some excess bandwidth will be used by the UDP flow, which causes the different bandwidth shares.

Figure 4.13. Bandwidth shares of three TCP flows using FIFO with tail-drop router
(1 CBR UDP flow + two TCP flows)

In the third experiment, we intend to examine if the SACC provides approximate fair max-min sharing properties, as in WFQ. Now there are three UDP flow (CBR) with rates of 0.5 Mbps, 1.2 Mbps and 2.0Mbps respectively, with the same RTT of 80 ms, competing for the 3Mbps bottleneck link bandwidth. The packet size is 256Bytes and network topology is the same as before. Figure 4.15 shows the bandwidth sharing among the three flows when FIFO with tail-drop discipline is applied. We can clearly see the shares of bottleneck bandwidth among the three flows are approximately 1: 2.12: 3.83. Figure 4.16 shows the bandwidth shares of the three flows when SACC with FDPS is applied. Now the shares of bottleneck bandwidth among the three flows become approximately 1: 2.35: 2.712. As a comparison, the ideal bandwidth sharing according to the fair max-min sharing policy is 1: 2.4 : 2.6. We can see that the behaviors of FDPS are close to that of WFQ in the pure best-effort environment.

Fig.4.14. Bandwidth shares of three TCP flows using FDPS in the router
(1 CBR UDP flow + two TCP flows)

Fig.15. Bandwidth shares of three UDP flows (CBR) using FIFO-Tail-drop in the router

In actual observations, the flow of rate 0.5Mbps experiences no packet drops and the flow with rate 1.2 Mbps experiences very few packet discards. In other words, most packets being dropped in the router by the FDPS belong to the flow of rate 2.0Mbps. That is, the more aggressive UDP flow with rate 2.0Mbps has been penalized by the SACC scheme by dropping its packets.

Figure 4.16. Bandwidth shares of three UDP flows (CBR) using FDPS in the router

4.5 Conclusions

We have presented a source aware congestion control (SACC) mechanism in this chapter. This is a shift in the paradigm in how congestion control/avoidance is provisioned – from transport layer to packet scheduling and buffer management layer. The proposed SACC scheme can deal with flows that are unresponsive to congestion notification or non-TCP-compatible flows which are responsive but more aggressive than TCP. Another notable feature of the SACC scheme is that it can gracefully work in conjunction with end-to-end congestion control mechanisms, such as TCP, and other

router assistant congestion control mechanisms, such as active queue management schemes.

This shift in paradigm in congestion control from TCP to a lower layer offers a new approach in which the source may partition its information into "must reach information" to be sent by conformant packets while "enhancement information" by non-conformant packets. The SACC mechanism uses the FDPS algorithm which allows the source to exercise greater control, as compared to TCP, over its connections even when congestion arises along its path. The FDPS algorithm provides packet priority marking, scheduling and discard mechanism and guarantees all conformant traffic and delivers non-conformant traffic in the best-effort manner.

We have shown how the SACC mechanism may work gracefully with TCP together with RED as the router assisted congestion control mechanism. We have shown that the SACC mechanism produces similar performance regardless whether RED is implemented or not, releasing the requirement of choosing appropriate parameter setting in RED.

Further investigation on the SACC mechanism in a best-effort environment shows that the TCP throughput dependence on RTT is eliminated by the FDPS algorithm. The effects of non-responsive flows are reduced significantly by discarding more of their packets and thus prevent TCP congestion collapse, a major threat in TCP networks. We have also shown that an approximate fair max-min sharing is achieved by means of the same algorithm when working in the pure best-effort environment.

Future work could assess the performance of mixed TCP-compatible and responsive but not TCP-compatible flows using the SACC mechanism. Other extensions to the work may include buffer dimensioning and pricing of network services.

CHAPTER 5

A NEW MULTISERVICE PROVISIONING MECHANISM WITH SERVICE CURVES ASSURANCE FOR PER-CLASS SCHEDULING DELAY GUARANTEES

Class-based traffic treatment frameworks such as Differentiated Service (DiffServ) have been proposed to resolve the poor scalability problem in the flow-based approach. Although the performance is differentiated in a class-based basis, how to provide delay guarantee for individual flows in the same class has not been well understood and the issue has not been adequately investigated. In this chapter, we present a new multiservices mechanism with Service Curves assurance for per-flow QoS deterministic guarantees. In this mechanism, Flow-state-dependent Packet Scheduling (FDPS) is employed to handle the different flows within the same class to provide an end-to-end class-based scheduling delay bound guarantee. With the fine granularity in this mechanism, we are able to provide a per-flow end-to-end delay bound independent of the number of intermediate nodes. We mathematically analyze the performance of a flow using this class-based FDPS and compare it with DiffServ to demonstrate the enhancements. We also use simulation experiments based on NS-2 to validate the proposed mechanism.

5.1 Introduction

The Internet has been traditionally designed to support the best-effort service model in which the network offers no assurance for the data delivery. This service model has

proven to be adequate for elastic applications, such as FTP, HTTP, and TELNET that tolerate packet delays and losses rather gracefully. With the rise in deployment of inelastic continuous media applications, the best-effort service model has become inadequate. To facilitate the co-existence of both the elastic and non-elastic applications, service differentiation and guarantee in terms of rate and/or end-to-end delay is highly desirable. To this end, the Integrated Services (IntServ) network architecture [BR94] was proposed to provide per-flow absolute QoS guarantee. IntServ is an interesting proposal for guaranteed service in which admission control is applied to every flow. Unless the bandwidth needed to support a new flow could be reserved in the selected path, the flow request would be rejected. However, it is often argued that the need for maintaining per-flow state and performing per-packet classification in routers seriously limits the scalability of this architecture and the lagging demand for per-flow absolute QoS has also restricted the deployment of IntServ. As a result, the interest in Internet QoS eventually shifted to the DiffServ architecture [BL98], which was proposed as a scalable solution based on the same paradigm as the Internet. It resolves these problems by offering per-hop behaviors as the service at each link and replace bandwidth reservation by capacity pre-allocation to different service classes. This architecture employs simple packet scheduling and buffer management at the core routers based on tags that are set at the edge routers. The edge routers are allowed to perform traffic management on a per-flow basis while the core routers are not. In this architecture, traffic entering a network is classified and conditioned at the network boundary, and is assigned to a small set of behavior (or flow) aggregates (also referred to as Per Hop Behaviors--PHB). This architecture achieves scalability by implementing complex classification and conditioning functions only at network boundary routers (which process lower volumes of traffic and lesser numbers of flows), and providing service differentiation inside the network for flow aggregates rather than on a per-flow basis [BL98]. Simple resource-management mechanisms--such as Random Early Detection (RED)[FL93]--have been proposed to provide service differentiation among traffic aggregates. Although some traffic management and packet marking mechanisms [SAH00, SU03] have been proposed for the edge routers, it is less flexible and provides weaker guarantees to flows.

In particular, it is non-trivial to provide per-flow rate or delay guarantees in this mechanism.

In DiffServ architecture, the QoS differentiation is made on a per-class basis. However, the implicit assumption is that the per-flow performance and the aggregate/class performance is close, which may not hold under certain circumstances [BE02]. This phenomenon calls for further analysis to identify quantitative deviations between the per-flow QoS and the class QoS so that applications can be made aware of the QoS levels to expect from the network. More importantly, understanding the dominant factors influencing the degree of deviation could help minimize the differences in the perceived performance among flows.

DiffServ is expected to be widely deployed across the Internet. The implementation of PHBs relies much on the scheduling and queueing schemes used in switches and routers. The scheduling schemes that could be implemented in DiffServ include priority queueing (PQ), weighted round-robin (WRR), PQWRR [HOU01], and class-based queueing (CBQ) [FL95]. CBQ controls the explicit rate for each traffic class by the rate control mechanisms at two schedulers: the general scheduler and the link-sharing scheduler [FL95]. Compared with PQ and WRR, PQWRR delivers the minimum delay and jitter for EF traffic and provides better bandwidth allocation for AF traffic and BE traffic by priority scheduling of EF traffic and non-EF traffic, and weighted round-robin scheduling of AF traffic and BE traffic.

Outside the context of DiffServ, there is also much work on packet scheduling for service differentiation and QoS provisioning, such as GPS, PGPS [PAR93], W2FQ [BE96], WRR [KAT91] etc. Most of these algorithms are focused on sharing the output bandwidth fairly among competing backlogged flows. In this family of scheduling algorithms, active flows which are not backlogged at the moment do not participate in the sharing and there is no memory of unconsumed service for any flow. These algorithms are mainly designed for best effort domain in which it is sufficient to consider fair bandwidth sharing. However, these algorithms are not appropriate in today's environment in which QoS flows may have different delay bounds. In other words, this scheme has to

schedule flows in such a manner that their rates and delay bounds are coupled. Considering the scalability limitation of per-flow based service provisioning, such as IntServ, a per-flow delay guarantee mechanism based on class-based scheduling is highly desirable.

As the traffic of a flow interacts with the traffic of other flows in a shared link, it is unlikely that its delay bound could be guaranteed unless the input traffic of each flow is bounded – the delay bound depends on the traffic arrivals and the capacity of the shared link. Further research works [CR91, CR91+, CR95] on bounding the traffic arrival follow. For delay bound guarantees, traffic regulators such as the leaky bucket are commonly used to limit the burstiness of a flow. This defines the arrival curve [CR91, CR91+] for the input traffic. The idea of packet scheduling using a packet finishing time together with a traffic regulator for each flow is a part of the rate control scheduling disciplines and a possible implementation in IntServ [BR94].

There are many debates on whether deterministic QoS should be provided in the Internet. Some researchers have argued that strict QoS adherence is not necessary. As the performance of the Internet is generally good except for some periods of time, it may not be necessary to provide deterministic QoS. We agree that this argument could be true for a large number of users but for a small number of users who have high value transactions such as money and investment brokers, we argue that deterministic guarantees equivalent to leased line service are necessary. Unless we could provide deterministic QoS guarantees, the probability to attract such high net-worth services to utilize the Internet is almost negligible. Therefore, the purpose of the current work is to provide a multiservice mechanism that provides deterministic QoS guarantee via end-to-end class-based scheduling delay bound guarantees together with best effort service.

Despite its importance, relatively little emphasis has been put on the understanding of the per-flow QoS under the per-class traffic treatment. A recent paper [SC03] has focused

on an admission control scheme which provides per-flow delay and bandwidth guarantees based on simple class-based priority queueing. The authors examine the basic properties of the worst-case behaviors in strict priority queueing systems using network calculus. Building upon these properties the flow admission control scheme is devised. Unfortunately, they have not provided an implementable mechanism for realizing the theoretical service curves. Therefore, practical algorithms for providing per-flow delay guarantee based on class-based treatment are yet to be studied. In [BE02], the authors investigate the Expedited Forwarding Per-Hop Behaviour (EF PHB) and its impact on worst case end-to-end delay jitter. They find that the worst case delay jitter can be made arbitrarily large, while each flow traverses at most a specified number of hops, if we allow networks to become arbitrarily large. They also propose some proven bounds on delay jitter for networks implementing the new definition of EF PHB, both in cases without loss and with loss.

It has been proven that an exact quantitative mapping between the per-flow and the class is a challenging issue [NI99]. In this chapter, we do not focus on the theoretical derivation of the relationship, but on a mechanism that realizes the mapping between the per-class performance and the per-flow performance. We try to provide per-flow delay performance guarantee in traffic aggregates. We employ the theory of Service Curves to provide the protection of traffic and traffic classifications based on queuing delay budget to provide end-to-end delay bound protection. The services are provided in such a manner that each flow is guaranteed the minimum service according to its delay requirement. This scheduling mechanism is subtly different from the WFQ [PAR93] scheme whose main focus is sharing the bandwidth fairly. In WFQ, a lower scheduling delay could be achieved by allocating a higher service rate to a specific flow at the expense of available bandwidth. Due to the way packets are scheduled in the proposed mechanism, a shorter delay bound could be provided to flows in low delay classes and longer delay bounds to flows in high delay classes without resorting to bandwidth over provisioning for a lower delay bound. In the proposed multiservice mechanism, we employ an interesting traffic classification based on queuing delay budget in which a flow is classified not according to its end-to-end delay requirement but its queuing delay budget. This facilitates more

efficient utilization of bandwidth. This new multiservice mechanism provides Service Curves assurance for per-flow QoS deterministic guarantees. The core component is a novel packet scheduling algorithm, called Flow-state-dependent Packet Scheduling (FDPS), which is employed to process different flows within the same class to provide an end-to-end class-based scheduling delay bound guarantee. With the fine granularity in this mechanism, we are able to provide per-flow end-to-end delay bound independent of the number of intermediate nodes. To the best of our knowledge, there is little known work that realizes the theory of Service Curves in practice. To complement the scheduling mechanism, we make use of Bandwidth Broker [QB] architecture to facilitate simple admission control so that new flow requests could be admitted with deterministic guarantees.

This multiservice mechanism could be implemented in a packet switching network or integrated into the Diffserv architecture to provide deterministic per flow QoS guarantees. We prove mathematically that the class-based scheduling delay bound for each flow is independent of the number of intermediate nodes. We also employ simulation experiments based on NS-2 simulator [NS] to validate the proposed mechanism and evaluate its performance.

The main contributions of this work are four fold: 1) an efficient packet scheduling mechanism that provides class-based scheduling delay bound; 2) a simple end-to-end queuing-delay-based flow classification method; 3) an admission control conditions test for flow admission and 4) an efficient provisioning mechanism for protecting flows for the deterministic QoS guarantees in a multiservice network.

The layout for the rest of the chapter is as follows. In the next section, we present the proposed multi-service mechanism for per-flow delay guarantee based on class-based traffic treatment. In particular, we propose a class-based FDPS algorithm as the core component of the multiservice mechanism. In Section 5.3, we present the mathematical analysis of the FDPS. In Section 5.4, we describe the control plane operations with a Bandwidth Broker for the proposed mechanism, with the emphasis on the admission control procedure. In Section 5.5, we provide the simulation results and analysis to validate the proposed mechanism and finally conclude this chapter in Section 5.6.

5.2 Multiservice Mechanism for Per-flow Delay Guarantee Based on Class-based Traffic Treatment

5.2.1 Overview of the Proposed Multiservice Mechanism

Our goal is to design an efficient mechanism for deterministic per-flow QoS provisioning based on class-based traffic treatment in a multiservice network with both streaming and elastic traffic. Its functions include classification of traffic flows, a scheduling-delay-based scheduling mechanism and an admission control procedure. The delay-based classification is one of the key features of the mechanism for end-to-end scheduling delay guarantees. Such classification allows for service differentiation [BL98] between flows belonging to different traffic classes. This method uses a set of scheduling delay bounds for the classification of flows. In more detail, flows are classified into their appropriate delay classes in such a way that the over-provisioning of bandwidth is minimized. This facilitates the realization of service curves for flows with different delay bounds. With different delay bounds defined for different traffic classes, a flow is classified not by its end-to-end delay requirement but by its queuing delay budget, different from current classifications commonly based on end-to-end delay requirements. Priority queuing discipline is applied among the different classes of backlogged traffic. The crux of the new mechanism is that we employ a new packet scheduling algorithm called Flow-state-dependent Packet scheduling (FDPS) to schedule the traffic within class. After a flow is classified, all its packets will be forwarded by the scheduler of that class. When its class is scheduled for service, this FDPS scheduler in that class schedules packets to the output for transmission. With this two-level hierarchical scheduling, we prove that per-flow delay guarantee can be provided based on class-based treatment.

Early research work on packet scheduling is mainly focused on sharing the bandwidth fairly among all backlogged flows. Unless the input traffic is bounded, it is not obvious how scheduling delay could be bounded. To bind the input traffic, a traffic regulator/policer is added to shape or identify the traffic to which this bound is adhered. This upper bound is defined by the *arrival curve* for the traffic in the theory of *Service Curves* [CR91, CR91+, CR95, SAR99, CR98]. Service Curves is a theoretical

mechanism for providing deterministic QoS guarantees at a link. It is applied in the FDPS mechanism that realizes the arrival and minimum service curves for delay bound guarantees. Due to the limitation of a single scheduling delay bound for all QoS flows in the FDPS [SI05] mechanism, a service differentiation process based on delay-based traffic classes is incorporated. In each traffic class, the FDPS algorithm is applied to extract the state of a flow in relation to each of its packets. The packets are then queued in its output buffer waiting for the class to be serviced. As the FDPS algorithm identifies out-of-profile traffic packets for the best effort service, these packets are forwarded to the lowest class in this mechanism – for service if there is no backlogged packet in queues of higher classes. In this chapter, we assume that there is sufficient buffer space at each node for in-profile packets and only out-of-profile or best effort packets are discarded in case of congestion. However, we will not discuss the buffering and packets discarding processes any further as these processes are beyond the scope of this chapter.

For deterministic QoS provisioning, admission control function at the ingress node is normally activated upon the arrival of a new flow connection request to verify that sufficient capacity is available to support this flow along the selected path without violating the QoS of existing flows; otherwise, it is rejected. After the packet scheduling algorithm determines the bandwidth required for the guaranteed QoS, this requirement is passed to the admission control function for reservation. We assume that a reservation protocol for aggregated bandwidth reservation is available. This would reduce the amount of signaling required to set up or tear down a connection, an area for further work. For the control panel operations of the proposed mechanism, we find that the Bandwidth Broker [QB] would be a suitable architecture for implementing admission control, bandwidth allocation, etc. Further details on the implementation will be given in Section 5.4.

In the following, we present the details on flow classifications, how capacity of a link is allocated to various traffic classes and how packets are scheduled at the class-based FDPS scheduler. A brief review of FDPS mechanism is also provided for the sake of completeness.

5.2.2 Classification of Flows

In the proposed multiservice mechanism, flow classification is performed at the ingress node in the connection establishment phase. Once a flow is classified, no further classification is made at other downstream nodes. This classification allows aggregation of flows and forwarding of packets belonging to the same class to a class-based scheduling mechanism. In contrast to typical classification by means of application type or end-to-end delay requirement, each flow is classified according to its queuing delay budget, denoted as d_i^q , which will be defined later. The purpose of this classification is to provide an end-to-end scheduling delay bound guarantee on a per-class basis. In other words, when a flow is classified into a specific class, its end-to-end scheduling delay is upper bounded by the scheduling delay of the class. This classification function assumes that the scheduling mechanism is able to provide a class-based end-to-end scheduling delay bound – the maximum scheduling delay due to a series of shapers and schedulers in a path for that class, independent of traffic conditions. This is an interesting result as it affects the manner flows with similar delay requirements are classified. For example, two voice connections between different end points would have different fixed delays due to different transmission and end-to-end propagation delays. As a result, these voice connections would have different queuing delay budget and would be classified into two different classes. In general, a flow is normally classified into a class with the highest scheduling delay bound smaller than or equal to its queuing delay budget. A formal definition for delay-based traffic classification is given below.

Definition 5.1: For all flows classified into class K with scheduling delay bound D_K , the end-to-end scheduling delay of every flow in this class is less than or equal to D_K .

The end-to-end delay of a flow could be separated into two components: a fixed delay that includes propagation and transmission delays; and a variable delay due to the effects of scheduling at each node along the path of the flow. By means of the proposed multiservice mechanism, the variable delay of every flow in class K is shown to be upper-

bounded by the scheduling delay D_K . The scheduling mechanism at every node along the path must ensure that the scheduling delay of the series of shapers and schedulers along the path is D_K . We leave the mathematical proof for this to part *E* of this section.

Let there be M traffic classes in the mechanism. Correspondingly, we specify M end-to-end scheduling delay bounds. The M delay bounds can be represented by a scheduling delay vector $\mathbf{D} = \{D_M, D_{M-1}, \dots, D_1\}$ at a link. Without any loss of generality, we assume $D_M < D_{M-1} < \dots < D_1$ with classes M and 1 representing the highest and lowest classes respectively. When a new flow connection request arrives, it is classified into one of the M classes in the connection establishment phase. As mentioned before, class 1 (the lowest priority class) is designated as the best effort class with unspecified delay bound. This ensures that best effort traffic does not interfere with QoS traffic in higher classes. With this delay vector $\mathbf{D}$ at each link, the traffic in a class with lowest delay bound should be serviced before the traffic in lower classes. Hence, backlogged packets at the highest class are always serviced before the rest. The following definition specifies how a flow with its path attributes is classified.

Definition 5.2: For a flow i with an end-to-end delay d_i and a fixed delay d_i^f , its queuing delay budget d_i^q is given by $d_i^q = d_i - d_i^f$. This flow i is then classified into class K such that

$$K = \max \{H : H \geq 1; D_H \leq d_i^q < D_{H+1}\},$$

subject to $C_{H,j} \geq R_i$

where $C_{H,j}$ is the available capacity of class H at link j and R_i is the service rate required by the new flow i request.

From this definition, we observe that flow i is classified into a class K with the largest scheduling delay bound not greater than the queuing delay bound of the flow i . In the event that there is insufficient capacity in class K , this flow could be admitted under class $K+1$ if there is available capacity resulting in a lower end-to-end delay bound of

$d_i^f + D_{K+1}$. This flow would then be classified into a class with lower delay bound than required due to a shortage of capacity in class K . The effect of capacity allocation to different traffic classes is presented next.

5.2.3 Link Capacity Allocation

With M classes defined in this mechanism, capacity allocation at a link affects the number of flows admissible to an individual class. We define ξ_j as a link capacity allocation vector for allocation at link j . This vector $\xi_j = [\alpha_M, \alpha_{M-1}, \dots, \alpha_1]$, where α_K is the fraction of the capacity allocated to class K and obviously, $\sum_{K=1}^M \alpha_K = 1$. This vector ξ_j represents the capacity allocation to different classes of traffic at a particular link j . A different vector ξ_j could be allocated at a link j according to a network operator's preference.

With the class-delay specification in Definition 1, traffic in higher classes (greater K) is scheduled before lower ones. The unconsumed service allocated to a higher class would roll over to the next lower class thus providing more capacity to this class. Unused service from all higher classes of traffic would be accumulated at the lowest class where best effort service is provided. This property could be exploited to provide service efficiently to users with different QoS requirement instead of relying on over-provisioning in WFQ family of scheduling to reduce queuing delay. The provisioning based on Service Curves assures the worst case provisioning. Hence, the queuing delay of a flow in this mechanism would be in general much smaller than its class scheduling delay bound. This is especially pronounced for flows in lower classes for link with large capacity. Such accumulation of unconsumed service rolled over to lower service class could be accumulated to class 2 to provide a guaranteed rate service with a large queuing delay bound. More discussions of this service are provided in Section 5.3.

For sake of simplicity, we assume that all links uses the same allocation vector ξ in this chapter. This vector ξ for M delay-classes in a network could be deemed as an abstraction in which there are M overlay virtual networks corresponding to the capacity of M classes at each link of the physical network. With this abstraction of M virtual networks, service at each link is provided to virtual network from the highest to the lowest – class M , followed by $M-1$, and then the rest sequentially. At the lowest class, the “Best Effort” virtual network comprises of unused capacity from higher classes. No capacity is allocated to the best effort class (Class 1).

In a class K virtual network, the capacity along the path of a flow is normally limited by a bottleneck link in that class. Admission control function would have to locate such bottleneck link along this path to determine whether there are sufficient resources to service a new flow. This admission control test is presented in Section IV. While a path capacity depends on the current utilization and its allocation vector ξ at its bottleneck link, the bandwidth required by a flow depends on the packet scheduling algorithm.

5.2.4 Packet Scheduling

Early research work shows that packet scheduling is one of the key issues for QoS guarantees. It is recognized that reserving the same capacity as a dedicated channel for a flow in a packet switching network is insufficient to provide the QoS guarantee to a flow due to the interference of other traffic sharing the links along its path. We have proposed a packet scheduling mechanism suitable for providing a class-based delay bound. It facilitates the implementation of service curves that provide different delay bounds for different classes of traffic. This mechanism consists of a set of schedulers as shown in Figure 5.1. Each class-based scheduler implements the FDPS algorithm that provides service curve assurance. Figure 1 depicts how a class K packet at the input buffer is forwarded to the class K scheduler, awaiting for service.

We recall that delay $D_K < D_{K-1}$ for all K , $1 \leq K \leq M$. To meet such scheduling delay requirements, packets from class K should be serviced before class $K-1$. It seems that

priority scheduling to these M classes of traffic in this scheme is a natural and reasonable choice. A total of M schedulers are need for these M classes of traffic. Each scheduler operates independently and only one scheduler is processing its packets at the output link at any time. Figure 5.1 shows how the class-based schedulers are activated from the highest class to the lowest class of traffic equivalent to strict priority scheduling. In other words, backlogged packets belonging to class K are always service before those of class $K-1$.

There are two levels of granularity in this scheduling mechanism – coarse and fine. At coarse granularity, the highest priority class with backlogged packets is always selected for service. This is similar to the static priority scheduling in DiffServ [BL98] in which the highest priority packets are always serviced first. At fine granularity, packets belonging to a class are scheduled by the FDPS algorithm [SI05]. When a class of traffic is selected for service, the FDPS algorithm schedules its packets to the output link. If packets from a higher class arrive, service is switched to this class. For example, if class K is the highest class with backlogged traffic, the class K FDPS algorithm is activated to schedule packets to the output link. However, if a new class $K+1$ packet arrives, the service is switched to class “ $K+1$ ” traffic using non-preemptive scheduling. Before we discuss the multiservice scheduling mechanism further, a review of the FDPS algorithm is provided.

Figure 5.1 The Multiservice Scheduling mechanism

5.2.5 The FDPS Algorithm

As mentioned above, the FDPS algorithm, which is a packet scheduling algorithm that provides Service Curves assurance, is applied to schedule packets in the same class for service. The FDPS implements an arrival curve and a service curve for each flow according to its traffic specifications and QoS requirements. It employs an enhanced token bucket (ETB) for fine granularity marking of packets from an individual flow. The ETB consists of two token buckets, namely, the positive and negative buckets. The positive bucket is dimensioned according to the traffic parameters of an individual flow. The algorithm uses the number of tokens in the positive or negative bucket to generate the priority for each packet. As an in-profile packet is identified by its positive dynamic priority, the traffic bounded by the arrival curve is easily identified. When the positive bucket is empty, negative dynamic priority is generated from the negative bucket. These negative priority packets constitute the out-of-profile traffic and form part of the best effort traffic in the network. Such packets are scheduled only if there is no backlogged in-profile packet.

This algorithm retrieves the state of a flow in relation to a packet to generate a dynamic priority for the purpose of scheduling. As the depth of the positive bucket is dimensioned by traffic parameters of a flow, the dynamic priority assigned encapsulates the state of a flow relative to its arrival curve. Each of these packets is then forwarded to the output queues according to the corresponding priority before priority based scheduling is applied to service these packets queued at the output queues. Please refer to Chapter 3 for details of the FDPS algorithm.

The main advantage of the multiservice mechanism is its ability to provide Service Curve assurance. One of the most interesting properties of service provisioning via Service Curves is the ability to reduce a series of shapers, service curve and delay elements into three equivalent elements in the network model. In [BOU01,CR91,CR91+], the resulting end-to-end elements delay due to Service Curve assurance can be computed by min-plus calculus, leading to the following theorem.

Theorem 5.1 For a series of V shapers, V service curves and V delay elements along the path of a flow, the series of elements could be represented by an equivalent shaper element $\bar{S}_{1V}$, an equivalent service curve element S_{1V} and an equivalent delay element $\delta_{d_{1V}}$. For a flow i passing through $\bar{S}_1(t), S_1(t), \delta_{d_1}(t), \dots, \bar{S}_V(t), S_V(t), \delta_{d_V}(t)$, where $\bar{S}_j(t), S_j(t), \delta_{d_j}$ are the arrival curve, service curve and delay element of node j , respectively. The three equivalent elements are given by

$$\bar{S}_{1V}(t) = \bar{S}_1(t) * \bar{S}_2(t) * \dots * \bar{S}_V(t)$$

$$S_{1V}(t) = S_1(t) * S_2(t) * \dots * S_V(t)$$

$$\delta_{d_{1V}}(t) = \delta_{d_1}(t) * \delta_{d_2}(t) * \dots * \delta_{d_V}(t)$$

and

$$\delta_d(t) = \begin{cases} 0 & \text{for } t < d \\ \infty & \text{for } t \geq d \end{cases}$$

Using the above theorem, we can reduce a series of network elements into three equivalent elements in the network model and thus simplify the design and analysis of the packet scheduling scheme. In particular, if identical shaper and service curve are used in every link of the path for a flow, the scheduling delay is equivalent to the service delay of one link allowing the simplification of the scheduling delay calculation. This has been applied to compute the scheduling delay for a flow and condition for admission control in [SI05]. To ensure that the service curve for each flow is provided, sufficient service capacity must be available to meet the minimum scheduling delay bound for all flows. This leads to small over provisioning for some of the flows with larger scheduling delay bounds. Theorem 1 is very useful for the proposed multiservice provisioning mechanism that implements the concepts of arrival and service curves. By grouping flows with similar queuing delay budget into the same traffic class, we are able to provide end-to-end delay bound guarantee on a per-flow basis.

5.3 Analysis of the Scheduling Mechanism

In the last section, we have discussed how a new packet arriving at a node is forwarded according to its class to a class-based FDPS scheduler for service. In this section, we present a mathematical analysis of the above mechanism which provides per-class end-to-end scheduling delay. While the FDPS algorithm provides Service Curves assurance, this analysis shows the multiservice scheduling mechanism provides per-flow QoS guarantees with Service Curves assurance with different scheduling delay bounds. To provide Service Curves assurance, we have to show that a pair of arrival and service curves is provided to each flow at every node.

5.3.1 Arrival Curve

We note that the multiservice scheduling algorithm forwards each packet according to its class to the FDPS scheduler of that class. This forwarding process provides service differentiation, followed by the operation of the FDPS algorithm that identifies the in-profile traffic for guaranteed service. When a packet is forwarded to its class-based

scheduler in Figure 5.1, the FDPS algorithm extracts the state of its flow for dynamic priority assignment by means of the Enhanced Token Bucket. As the positive bucket is dimensioned by the traffic parameters of the in-profile traffic of its flow, this packet released by this bucket represents conformant traffic of the flow. The FPDS algorithm computes the dynamic priority from the number of tokens in the positive bucket. In the event that a flow is non-conformant, the positive bucket is empty and the negative bucket is accessed for negative priority assignment. The FPDS algorithm separates conformant packets from non-conformant packets by this priority assignment procedure which leads to the following lemma.

Lemma 5.1: All packets of a flow i which are marked with a positive dynamic priority belongs to the traffic bounded by its arrival curve $\bar{S}_i(t)$.

Proof: This result follows directly from the proof in [SI05]. Due to the manner the arrival curve of a flow is parameterized, all packets with positive dynamic priority constitute traffic under the arrival curve. The contrary is also true: all negative priority packets of a flow i represents the traffic not bounded by the arrival curve $\bar{S}_i(t)$. Such traffic would be forwarded to best effort class for service. This completes the proof.

5.3.2 The Service Curve

The FPDS algorithm provides the minimum service curve to a flow in conjunction with an associated admission control function. After defining the arrival curve $\bar{S}_i(t)$ for flow i , its minimum service curve $S_i(t)$ is given by $S_i(t) = \bar{S}_i(t - D_{\min})$, where $D_{\min}$ is the minimum scheduling delay bound, among flows sharing a link j . While some flows may require such a low scheduling delay, many do not. This leads to over provisioning for many flows in which more bandwidth is allocated than necessary. The multiservice scheduling mechanism is proposed to overcome this over provisioning. Notice that a scheduling delay vector $\mathbf{D} = \{D_M, D_{M-1}, \dots, D_1\}$ is defined for M classes of traffic for the network in Section II. A flow is normally classified into its most appropriate class whose delay is not

greater than its queuing delay budget. The queuing delay of a flow is thus bounded by the scheduling delay bound of its traffic class, as stated in the following lemma.

Lemma 5.2 For a flow i in class K in a network implementing the multiservice mechanism, its actual queuing delay d_i^a is bounded by the scheduling delay D_K of its traffic class if and only if $\bar{S}_K(t - D_K) \leq S_K(t)$, where $\bar{S}_K(t)$ and $S_K(t)$ represent the aggregate arrival and service curves at the output link.

Proof: For a flow i in class K , we have to prove that its actual queuing delay $d_i^a \leq D_K$. Consider h flows in class K at a link j , let $\bar{S}_i(t)$ be the arrival curve for flow i at this link. The arrival curve of the aggregated traffic of class K is then given by $\sum_{i=1}^h \bar{S}_i(t)$. For clarity in notation, we use $\bar{S}_K(t)$ and $S_K(t)$ to denote the aggregate arrival and service curves for class K traffic. From the theory of Service Curves [BOU01,CR91,CR91+], if the aggregate arrival curve for scheduling delay bound D_K is given by $\bar{S}_K(t) = \sum_{i=1}^h \bar{S}_i(t)$ and aggregated minimum service curve is $\sum_{i=1}^h \bar{S}_i(t - D_K)$. If $\bar{S}_K(t - D_K) \leq S_K(t)$, the scheduling delay of class K traffic is bounded by D_K . This completes the proof.

Note that d_i^a refers the actual queuing delay while d_i^q in Section II refers to the queuing delay budget. The following must hold in this mechanism: $d_i^a \leq D_K \leq d_i^q$. By providing this service curve $S_K(t)$, D_K is the maximum packet scheduling delay of traffic class K . For the provisioning of minimum aggregate service curve $S_K(t)$ for class K , the admission control function at the appropriate ingress node would have to verify that the above condition is true for all links in this path of a new flow before it is admitted. A simple admission control test is formulated for admission of new flows. This issue is reduced to verifying whether this service curve could be provided at the bottleneck link of the path.

5.4 Admission Control

In this section, we present some control plane operations in a Bandwidth Broker (BB) architecture for the proposed multiservice mechanism. We first give a brief architectural overview of a BB and then discuss the bandwidth reservation as well as the admission control procedure.

5.4.1. An overview of Bandwidth Broker

In the IETF DiffServ framework, a centralized model based on the notion of bandwidth Broker (BB) [QB] has been proposed for the control and management of QoS provisioning. In this centralized model, each network domain has a BB which is implemented as a special network server. The BB is responsible for maintaining the network QoS states and performing various QoS control and management functions such as resource reservation, admission control, and provisioning for the entire network domain.

Similar to that in [HOU01], the BB architecture relies on the packet classifier and the ETB in FDPS schedulers to provide a QoS abstraction in the data plane. Each router in the network domain employs the class-based FDPS scheme. All QoS reservation and other QoS control state information (e.g., the amount of bandwidth reserved at a core router) is removed from core routers, and is solely maintained at and managed by the BB. In supporting the different class of QoS and the best effort services in a network domain, core routers perform no QoS control and management functions such as admission control, but only data plane functions such as packet scheduling and forwarding. The BB centrally maintains and manages a number of management information (data) bases regarding the network domain. The BB consists of several modules such as admission control, QoS routing, and policy control. In the BB model, the network QoS states are represented at two levels: link-level and path-level. The link QoS state database maintains information regarding the QoS states of each link in the network domain, such as the total reserved bandwidth or the available bandwidth of the link. The path QoS state database maintains the QoS state information regarding each path of the network domain,

which is extracted and “summarized” from the link QoS states of the links of the path. An example of the path QoS state is the available bandwidth along a path, which is the minimal available bandwidth among all its links. By maintaining a separate path level QoS state, the BB can conduct a fast admissibility test for flows routed along the path. Both the link QoS states and path QoS states are aggregate QoS states regarding the links and paths. No per-flow QoS states are maintained in either of the two QoS databases—the QoS and other control state information regarding each flow such as its QoS requirement and reserved bandwidth is maintained in a separate flow information database managed by the BB.

5.4.2. Bandwidth Reservation

In this Multiservice provisioning mechanism, we assume that a route selection process is available to select the end-to-end path for the admission control test in the BB. The BB would ensure that sufficient capacity is available to meet the QoS of all flows before a new flow is admitted.

For bandwidth reservation along the path, we would like to reserve the minimum bandwidth to support a flow request so that more flows could be admitted. This can be done using the above mentioned BB architecture. The traffic parameters of a flow allow the definition of its in-profile traffic for which QoS provisioning is needed. We assume that a large pipe reservation is available for bandwidth reservation in the BB architecture. Such reservation is less frequent than per-flow reservation as unused capacity can be used for future connections. However, such reservations would require more background management operations to reallocate bandwidth when the unallocated bandwidth becomes depleted. The study on the large pipe reservation protocol is beyond the scope of this chapter. However, such large pipe reservation reallocation could be implemented efficiently in the BB architecture as all QoS states are maintained in the BB information (data) base. In this architecture, all link-state-information are stored in the BB. When a new flow starts or an existing flow terminates, signaling between the edge node and BB updates the BB of such changes. The BB, in turn, updates the changes in link-state-

database. Once the decision to admit a flow is made, the BB would send a confirmation to ingress router and the routers in the selected path and set traffic policing and scheduling polices for the flow. This completes the bandwidth reservation of the control panel operation involving the BB.

5.4.3 Admission Control Analysis

As discussed above, an efficient admission control test and resource allocation can be performed by the BB in conjunction with the class-based FDPS scheme in the proposed mechanism. Unlike the conventional hop-by-hop approach which performs admission control individually based on the local QoS state at each router along a path, we employ a path-oriented approach to examine the resource constraints along the entire path that the new flow will traverse, and make admission control decision accordingly. As a result, we can significantly reduce the time of conducting admission control test. Clearly, such a path-oriented approach is possible because the availability of QoS state information of the entire path at the BB.

In Section 5.3, we discuss the requirement for service provisioning to ensure that the minimum aggregate service curve could be provided to a class of traffic. In the following part, we would derive the conditions under which the aggregated service curves for the M classes of traffic could be provisioned. As discussed in Section 5.2, we apply priority scheduling to schedule the backlogged traffic of the class with the shortest delay first. Such priority scheduling ensures that the delay of the higher class is always lower than lower classes and the traffic in higher classes would not be affected by the traffic in lower classes and leads to the following lemma.

Lemma 5.3 For M classes of traffic in a network implementing the multiservice mechanism, a new flow request i belonging to class L is admitted if and only if

$$\frac{\sum_{N=K}^M \sigma_N}{D_K} \leq C_j - \sum_{N=K+1}^M \rho_N \quad K = 1, \dots, M-1$$

$$1 \leq L \leq M$$

where

$$\rho_N = \sum_{i \in F_N} \rho_{N,i} \quad i \in F_N,$$

$$\sigma_N = \sum_{i \in F_N} \sigma_{N,i} \quad i \in F_N,$$

$$\sum_{N=H}^M \rho_N = 0 \quad \text{if } M < H$$

where $\sigma_{L,i}$ is the bucket depth of flow i in class L , F_L is the set of all flows in class L and C_j is the capacity of link j .

Proof: Different classes of traffic in the multiservice mechanisms are considered below:

For class M ,

$$\bar{S}_M(t) = \sum_{i \in F_M} (\sigma_{M,i} + \rho_{M,i}t) \quad i \in F_M$$

To guarantee the minimum aggregate service curve, we require

$$\bar{S}_M(t - D_M) \leq S_M(t)$$

For $S_M(t) = C_j t$ and $t = D_M$, we have

$$\sum_{i \in F_M} \sigma_{M,i} \leq C_j D_M \quad i \in F_M$$

Thus, the required condition is

$$\frac{\sigma_M}{D_M} \leq C_j$$

For class K , we have

$$\bar{S}_K(t - D_K) \leq S_K(t)$$

Considering all traffic classes above class K together with class K traffic, we have

$$\sum_{N=K}^M \bar{S}_N(t - D_N) \leq C_j t$$

Note that $D_M < D_{M-1} < \dots < D_1$ in the network. For $t = D_K$, we have

$$\sum_{N=K+1}^M (\sigma_N + \rho_N D_K) + \sum \sigma_K \leq C_j D_K$$

To satisfy the service curve of all flows including the new flow, the following condition must be satisfied,

$$\frac{\sum_{L=K}^M \sigma_L}{D_K} \leq C_j - \sum_{L=K+1}^M \rho_L \quad K = 1, 2, \dots, M$$

$$1 \leq L \leq M$$

An intuitive interpretation of this inequality is as follows. The right-hand-side (RHS) denotes the residual capacity of a link after deducting the long-term rate of the traffic above class K , the result of priority scheduling. The LHS of the inequality denotes the minimum capacity for a scheduling delay bound of D_K . Obviously, the residual capacity must be greater than this minimum requirement if the delay bound D_K is satisfied.

5.4.4 Effects of Capacity Allocation

We recall in Section II that the capacity allocation at a link could be defined by an allocation vector $\xi_j = \{\alpha_M, \alpha_{M-1}, \dots, \alpha_1\}$. Given this allocation vector ξ_j , we derive a simple admission control test for admission control of a new flow. As pre-allocation of capacity to different traffic classes is reserving capacity for these classes, some unused capacity would result. Unused capacity in a higher traffic class is rolled over to the next lower class till it reaches class 1, the “best effort” class. We note that the RHS of inequality in Lemma 3 could be replaced by the allocated capacity of their classes. If capacity pre-

allocation is applied, the RHS of this inequality could be replaced by the capacity reserved for their classes. This action reserves capacity for different classes even though there could be no traffic in some of these classes at the moment. We propose the following theorem for capacity allocation.

Theorem 5.2 For the given allocation vector $\xi_j = \{\alpha_M, \alpha_{M-1}, \dots, \alpha_1\}$ for link j with capacity C_j , a new flow request i is admitted to class K if and only if

$$\sigma_K \leq C_j \left\{ \left(1 - \sum_{N=K+1}^M \alpha_N\right) D_K - \left(1 - \sum_{N=K+2}^M \alpha_N\right) D_{K+1} \right\} \quad \text{for } K \leq M$$

and

$$\rho_K \leq C_j \alpha_K$$

where

$$\rho_K = \sum_{\forall i} \rho_{K,i}$$

$$\sum_H^M \alpha_N = 0 \quad \text{if } M < H$$

and

$$D_{M+1} = 0$$

Proof: By pre-allocation of link capacity to M classes, the capacity $C_j \alpha_K$ is reserved for class K at link j . For class M , the following inequalities must hold:

For Class M traffic, we have

$$\frac{\sum_{\forall i} \sigma_{M,i}}{D_M} \leq C_j \quad i \in F_M$$

$$\sum_{\forall i} \rho_{M,i} \leq C_j \alpha_M$$

where F_M is the set of all flow in class M .

Therefore

$$\sigma_M \leq C_j D_M; \quad \sigma_M = \sum_{\forall i \in F_M} \sigma_{M,i}$$

$$\rho_M \leq C_j \alpha_M; \quad \rho_M = \sum_{\forall i \in F_M} \rho_{M,i}$$

For class $M-1$ traffic, we have

$$\frac{\sigma_M + \sigma_{M-1}}{D_{M-1}} \leq C_j - \rho_M$$

But $\rho_M \leq C_j \alpha_M$ and $\sigma_M \leq C_j D_M$, we have

$$\sigma_{M-1} \leq C_j (1 - \alpha_M) D_{M-1} - C_j D_M = C_j [(1 - \alpha_M) D_{M-1} - D_M]$$

By induction, we have

$$\sigma_K \leq C_j \left\{ \left(1 - \sum_{N=K+1}^M \alpha_N\right) D_K - \left(1 - \sum_{N=K+2}^M \alpha_N\right) D_{K+1} \right\} \quad \text{for } K \leq M$$

$$\rho_K \leq C_j \alpha_K$$

$$\text{where } \sum_H^M \alpha_N = 0 \quad \text{for } M < H \quad \text{and } D_{M+1} = 0$$

The proof is completed.

Some interesting observations could be made from this set of inequalities in Theorem 5.2. There are two constraints that a class of flows is subjected to. The two different tests for a flow at admission control are burstiness and long-term rate as reflected in the first and second inequalities. For constant rate flows of a particular class k , the number of flows is limited by the aggregated rate bounded by rate $C_j \alpha_k$. For bursty flows at class k , the number of flows is limited by the aggregated burstiness bounded by

$$C_j \left\{ \left(1 - \sum_{N=K+1}^M \alpha_N\right) D_K - \left(1 - \sum_{N=K+2}^M \alpha_N\right) D_{K+1} \right\} \quad \text{for } K \leq M$$

With these results, we could assign constant rate flows up to an aggregate rate of $C_j \alpha_K$ for class K with scheduling delay bound D_K , an elegant way for providing constant rate service with bounded scheduling delay.

For end-to-end admission control, the capacity of a path is thus bounded by the maximum burstiness and rate allocation of its class for all links in the path of its flow.

Therefore,

$$\begin{aligned} \sigma_K &\leq C_j \left\{ \left(1 - \sum_{N=K+1}^M \alpha_N\right) D_K - \left(1 - \sum_{N=K+2}^M \alpha_N\right) D_{K+1} \right\} & \text{for } K \leq M, j \in L_i \\ \rho_K &\leq C_j \alpha_K \end{aligned}$$

where L_i is the set of all links in a path of a flow i .

Hitherto, we have presented a mechanism in which the queuing delay of a flow is bounded by the scheduling delay of its class by providing the same arrival and service curves for a particular flow at every link in its path. By applying the theory of Service Curves, its end-to-end class-based scheduling delay bound could be computed elegantly. We have also derived a simple admission control test for a network which pre-allocates capacity to different classes of traffic at a link. In the following, we give examples to illustrate the operation of the admission control in this multiservice mechanism.

5.4.4 Admission Control Examples

We provide two examples to illustrate how delay-based traffic classifications could increase the number of flows admitted with deterministic QoS. In the first example, we consider a simple case in which there are two classes of traffic with scheduling delay bounds of 20ms and 80ms, respectively. A link with 5 Mbps is shared by a number of video flows with the parameters shown in Table 5.1.

TABLE 5.1 FLOW PARAMETERS

Parameter	value
ρ_i	0.5 Mbps
σ_i	40 Kbps
Average packet length $\bar{l}$	4 Kbits

We consider two cases in this example: a one-class of 20ms; and a two-class of 20 ms and 80 ms respectively.

Case 1: A one-class mechanism. In this case, the whole link capacity is available to carry traffic with 20ms scheduling delay bound. Let n be the number of admissible flows. Applying Theorem 5.2, we have

$$\frac{n \times 40 \times 10^3}{20 \times 10^{-3}} \leq 5 \times 10^6 \Rightarrow n \leq 2$$

$$n \times 0.5 \times 10^6 \leq 5 \times 10^6 \Rightarrow n \leq 10$$

Therefore, $n = 2$, two flows could be admitted in this case.

Case 2: A two-delay-class mechanism. In this mechanism, two classes are defined. We assume that the capacity of the link is allocated equally between the two classes. Let n_2 and n_1 be the number of flows admissible for classes 2 and 1 respectively. Note that Class 2 has shorter scheduling delay bound of 20ms. For this case, $\xi_j = [\alpha_2, \alpha_1]$, where $\alpha_2 = 0.5$ and $\alpha_1 = 0.5$

Applying Theorem 5.2 to class 2 traffic, we have

$$\frac{n_2 \times 40 \times 10^3}{20 \times 10^{-3}} \leq 5 \times 10^6 \Rightarrow n_2 \leq 2$$

and $n_2 \times 0.5 \times 10^6 \leq 5 \times 10^6 \times 0.5 \Rightarrow n_2 \leq 5$

We thus have $n_2 = 2$, two flows at class 2 with scheduling delay bound of 20ms could be admitted.

Applying Theorem 5.2 to class 1 traffic, we have

$$n_1 \times 40 \times 10^3 \leq 5 \times 10^6 [0.5 \times 80 \times 10^{-3} - 20 \times 10^{-3}] \Rightarrow n_1 \leq 2.5$$

and $n_1 \times 0.5 \times 10^6 \leq 5 \times 10^6 \times 0.5 \Rightarrow n_1 \leq 5$

We thus have $n_1 = 2$, two flows at class 1 with 80ms scheduling delay bound could be admitted.

In this example, only two flows are admitted if one scheduling delay bound of 20ms is defined while a total of 4 flows could be admitted if two scheduling delay bounds of 20 ms and 80 ms are defined, illustrating the advantage of specifying different scheduling delay bounds for different classes.

In the second example, we consider a case in which there are three classes of traffic with scheduling delay bounds of 20ms, 80ms and 320ms, respectively. A link with 100 Mbps is shared by a number of video flows with the parameters shown in Table 5.1.

Now there are three classes of traffic in the multiservice QoS mechanism. We assume that the capacity of the link is divided equally among the three classes. Let n_3 , n_2 and n_1 be the number of flows admissible for classes 3, 2 and 1 respectively. Note that Class 3 has shortest scheduling delay bound of 20ms. For this case, $\xi_j = [\alpha_3, \alpha_2, \alpha_1]$, where $\alpha_3 = 0.33, \alpha_2 = 0.33, \alpha_1 = 0.33$

Applying Theorem 5.2 to class 3 traffic, we have

$$\frac{n_3 \times 40 \times 10^3}{20 \times 10^{-3}} \leq 100 \times 10^6 \Rightarrow n_3 \leq 50$$

$$\text{and } n_3 \times 0.5 \times 10^6 \leq 100 \times 10^6 \times 0.33 \Rightarrow n_3 \leq 66$$

Therefore, $n_3 = 50$, 50 flows at class 3 with scheduling delay bound of 20ms could be admitted.

Applying Theorem 5.2 to class 2 traffic, we have

$$n_2 \times 40 \times 10^3 \leq 100 \times 10^6 [0.67 \times 80 \times 10^{-3} - 20 \times 10^{-3}] \Rightarrow n_2 \leq 84$$

$$\text{and } n_2 \times 0.5 \times 10^6 \leq 100 \times 10^6 \times 0.33 \Rightarrow n_2 \leq 66$$

Therefore, $n_2 = 66$, 66 flows at class 2 with 80ms scheduling delay bound could be admitted.

Applying Theorem 5.2 to class 1 traffic, we have

$$n_1 \times 40 \times 10^3 \leq 100 \times 10^6 [0.34 \times 320 \times 10^{-3} - 0.67 \times 80 \times 10^{-3}] \Rightarrow n_1 \leq 138$$

$$\text{and } n_1 \times 0.5 \times 10^6 \leq 100 \times 10^6 \times 0.33 \Rightarrow n_1 \leq 66$$

Therefore, $n_1 = 66$, 66 flows at class 1 with 320ms scheduling delay bound could be admitted.

In the second example, a total of 182 flows could be admitted if three scheduling delay bounds of 20 ms, 80 ms and 320ms are used. These two examples illustrate the benefits of having multiple scheduling bounds in the proposed mechanism. For a link with very large capacity, say 1 Gbit/s, many traffic classes could be defined to admit more flows according to their scheduling delay bound requirements. Suppose two video flows with delay requirement of 400ms and have fixed delay overheads of 50 and 300ms, these flows could be admitted to class 1 (320ms) and 2 (80ms) respectively without violating their end-to-end delay in this multiservice mechanism. We have shown that flows with

the same QoS requirement (400ms delay) admitted to different classes according to their queuing delay budget discussed in Section 5.2.

5.5 Validation of the Proposed Multiservice Mechanism by Simulation

In this section, we validate the proposed multiservice mechanism for per-flow delay guarantee based on class-based traffic treatment through simulation experiments based on network simulator NS-2 [NS].

We consider a network with two edge routers and one core router, as shown in Figure 5.2. There is a bottleneck link between the core router and the second edge router, which has 5.3Mbps bandwidth and 5ms delay. As a result, congestion and thus packet discard are likely to occur at this link. All other links have 10 Mbps bandwidth and 5 ms delay.

In the first experiment, there are 3 classes traffic: class 2 with highest priority, class 1 and best effort. Both class 2 and class 1 have two VBR video flows respectively. The best effort traffic has one TCP flow. The scheduling delay bounds for class 2 and class 1 traffic are 20ms and 80 ms respectively.

Table 5.2 EXPERIMENT PARAMETERS OF THE TOKEN BUCKET OF THE FDPS SCHEDULER

Parameter	Video Traffic	TCP Traffic
Token Rate (bps)	0.5	0.5 M
Depth (bit) σ	40000	8000
Queue length	100packets	100 packets
Reserved rate ρ	0.5Mbps	0.5Mbps
Average packet length	4000 bits	8000bits

The parameters of the token bucket for video flows and TCP flow used in the simulation experiments are given in Table 5.2. Each simulation lasts for 60 seconds unless otherwise specified.

Figure 5.2 Network topology used in the simulation experiments

Figure 5.3 depicts the delay statistics of the three classes of traffic in experiment 1. We can see that the class 2 has the shortest maximum delay of 18.2 ms, class 1 has maximum delay of 37.5ms, while the TCP flow which uses best effort service has maximum delay of about 61ms. Therefore, the delay bound adherence according to Theorem 2 in the proposed mechanism is realized. In this experiment, no packet loss is experienced.

Figure 5.3 Comparison of CCDF for scheduling delays of class 1, class 2 and beset effort traffic (Experiment 1)

In the next experiment, we keep the network topology but an additional class 2 flow is admitted. This flow has the identical parameters with the previous class 2 video flow. That is, there are three video flows in class 2. Figure 5.4 shows the new results. We can observe that the scheduling delay bound of 20ms is violated as predicted by admission control. This result confirms our analysis in the admission control. We can observe that the queuing delay of most packets of class 2 and 1 traffic are much lower than the scheduling delay bounds of the respective classes as discussed in Section 5.2. The bandwidth allocation is based on worst case scenario for all flows using the network—that is, all flows start their bursty transmission in a synchronous manner.

Figure 5.4 Comparison of CCDF for scheduling delays of class 1, class 2 and beset effort traffic (Experiment 2: an additional class 2 video flow is admitted)

In the third experiment, the network configurations are identical to those in experiment 1, except that we now set one of class 2 video flows to consist of non-conformant traffic. We assume the non-conformant part of packets have been marked by FDPS and thus re-directed to the best-effort class in the operation of class-based FDPS. According to the class-based FDPS operations, this specific flow may experience packet loss due to the burstiness of the traffic at the higher classes. Figure 5.5 shows the packet loss statistics during 60 seconds time.

In this experiment, the total number of received packets is 5321 while 21 packets are lost. The packet loss due to violation of scheduling delay bound shows that though class 2 traffic is affected, class 1 traffic is unaffected due to its higher scheduling delay bound. More best effort packets are discarded due to the overloaded condition.

Figure 5.5 Packet loss when one class 2 flow includes non-conformant traffic

Through these simulation experiments, we can see that the proposed multiservice mechanism is able to provide per-flow delay guarantee via class-based traffic treatment, by means a simple admission control process.

5.6 Conclusions

In this chapter, we have proposed a novel multiservice mechanism for class-based end-to-end scheduling delay bound. Flows are classified according to their queuing delay budget rather than end-to-end delays. With different scheduling delay bounds for different traffic classes, each flow is classified into the lowest class that meets its queuing delay budget. Over-provisioning is minimized for efficient network utilization and admitting more guaranteed flows. We have proved mathematically that its scheduling mechanism provides Service Curves assurance to individual flows for deterministic QoS guarantee. We have also presented a simple admission control test in this multiservice mechanism using the BB architecture for deterministic service provisioning. Through

simulation experiments, we have verified that the mechanism is able to provide a class-based scheduling delay bound independent of the cross traffic behaviors. Due to the simplicity of this mechanism, it could be implemented independently in routers or be integrated into the DiffServ architecture as an enhancement for deterministic QoS guarantees. Further work on this mechanism could include stochastic QoS guarantees for the traffic class next to the best effort class. Pricing could be investigated to maximize the profit of the network operator using different charging schemes for different types of deterministic QoS guarantees.

CHAPTER 6

A MULTISERVICE FRAMEWORK BASED ON THE FDPS ALGORITHM

In this chapter, we propose a new Multiservice (MultiServ) framework based on the FDPS algorithm. It addresses the issue of provisioning deterministic service for different queuing delay bounds in a multiservice network. Recognizing that IntServ provides deterministic QoS but has unresolved scalability and complexity issues, and DiffServ provides coarse QoS guarantees, requirements of a group important users who require leased line service are still unaddressed. Unless an “equivalent lease line service” is available in the next generation network (NGI), integration of all services onto the NGI would seem unlikely.

6.1 Introduction

In Chapter 2, we have mentioned that the traffic in the Internet can be roughly classified as streaming or elastic traffic [SH95, RO04+, FI02, K004]. The former refers to the traffic with specific QoS requirements. We could further divide these requirements into deterministic and statistical QoS guarantees. Deterministic guarantee implies that the network would guarantee service rate, delay bound and loss (no loss) to a flow independent of the prevailing traffic condition. This guarantee provides a strict protection to the traffic based on the worst-case design. Examples of the traffic with deterministic QoS requirements include financial transaction of money brokers, remote surgical operations that utilize leased lines for end-to-end transmission, etc. For such applications, the failure in QoS guarantee can result in financial loss. On the other hand, the latter refers to a probabilistic guarantee of rate, delay and packet loss rate to a flow. Such a

service is normally acceptable to applications which can tolerate small amount of packet losses. It is recognized that statistical guarantee would facilitate more efficient bandwidth utilization [KN95, KN99, EI93] of the network. Rather than catering to a worst-case design, it assumes a predefined small probability that packet is lost or delayed. Applications that could tolerate some losses include voice and video services.

Elastic traffic refers to traffic from applications that can adapt to changing network conditions. An end-to-end protocol at a higher layer ensures the reliability of information transferred. The network would provide a best effort service to this type of traffic without delivery guarantee. Most traffic supported by TCP protocol belongs to elastic traffic.

Central to this multiservice framework is the provision of a so-called “equivalent leased line service (ELLS)” in a network. While many are familiar with virtual circuit service in packet switching and virtual leased line service [NI99] in DiffServ, we define concept of a new service, “Equivalent Leased Line Service (ELLS)”. In ELLS, a user would have the same experience whether he is using a leased line service or an “equivalent leased line service” in a packet switch network. More specifically, he would experience the same service rate, delay and loss (no loss) characteristics in either of these services. From the user perspective, he could not differentiate the services provided by this ELLS and a physical leased line (dedicated channel).

The natural question may arise, what is the difference between virtual lease line in DiffServ and ELLS? From the paper [BE02], we recognize that DiffServ provides only a probabilistic guarantee to an arbitrary delay bound of a flow. The packet delay could exceed this bound under specific conditions. Whereas, in ELLS, we provide a guaranteed rate and hard delay bound to a flow. We apply the theory of service curves to guarantee the quantitative delay bound. While we recognize that Service Curves theory assumes a worst-case design, a class-based treatment of flows would be more efficient in bandwidth utilization and the complimentary best effort traffic in the background could make efficient utilization of the network.

An incredible amount of research work related to QoS provisioning over the last decade has resulted in a number of QoS frameworks or architectures [KATS97, CHU00, AG98, BR94, BL98]. We here briefly review two well-known QoS architectures: Integrated Service (IntServ) [BR94] and Differentiated Service (DiffServ) [BL98]. The purpose of the former is to provide per-flow QoS guarantee in the Internet paradigm. Two of the main building blocks for IntServ are packet scheduling (by weighted fair queuing) and bandwidth reservation with admission control. It utilizes the RSVP [WR97] protocol for resource reservation along the path of a flow request before it is accepted. It requires maintenance of per-flow state at every node and frequent update of these states. For WFQ scheduling, it computes the packet finishing time for each packet to determine the order of service leading to a computational complexity of $\log(N)$ where N is the number of flows. The Internet community recognized that computation complexity and frequent updating of states would be the scalability issue that limits the number of flows at each link. To address the issue of scalability, Blake [BL98] *et.al* propose the DiffServ architecture for service differentiation and coarse QoS provisioning. It addresses the issue of scalability by means of flow aggregation. It recognizes that concentration of large number of flows at core nodes would cause scalability issue for per-flow states maintained at each of the core nodes. Its philosophy is to move the complexity to the edge nodes, which process less flows and keep the core nodes simple. Specifically, DiffServ has a limited number of traffic classes for flow classification. Having this property and applying priority scheduling to service the packets, it provides an elegant solution to the scalability issue in IntServ. However, DiffServ does not include admission control mechanism in the architecture. Admission control allows the network to control the admission of new flows in order to protect the QoS of existing flows. Without this function, DiffServ provides service differentiation between different classes of traffic and a coarse QoS if sufficient capacity is available in each path. Thus, DiffServ provides traffic flows with a per-hop per-aggregate service and its end-to-end characteristic is the convolution of these per-hop per-aggregate behaviors in its path.

We recall from Chapter 2 that it is not possible to provide protection to flows unless admission control is available. The Bandwidth Broker (BB) architecture for admission

control in DiffServ is proposed in [QB]. It complements the DiffServ architecture by keeping the core nodes simple but keeping stateful information at the BB. In this enhanced architecture, each DiffServ domain has one bandwidth broker. Each BB acts as the centralized controller in which an edge node interacts for the purpose of admission control. It maintains a QoS-state database of all link states and thus allows stateful admission of flows without the interacting with core node to get the latest state information. The problem of erroneous admission decision due to obsolete state information caused by delayed updated is resolved in this architecture. In other words, the BB architecture allows per flow admission control without the complexity that affects distributed architectures (e.g. RSVP) in which each node along its selected path takes part in the reservation. By means of the QoS state database in the BB, it performs a stateful admission control decision for each flow within its domain.

An important feature of the BB architecture is its ability to aggregate flow requirements when flows span more than one DiffServ domain. A BB can aggregate requests for resources from different flows for each adjacent domain before making an aggregate request to the domain [MAN04]. The relevant BB would aggregate flow termination requests and updates the QoS state database. For multi-domain connection, the BB in the source domain would send the aggregated information to all BBs involved. This provides a scalable approach to inter-domain connections reducing the amount of signaling for inter-domain requests.

In [QB], the ingress node selects a path for a new flow before it sends a connection request to the BB for admission control processing. The BB provides an elegant solution to control the amount of traffic in the premium service (expedite forwarding service – EF) in a DiffServ network for QoS guarantee. In [NI99], Nichols *et.al.* describe the premium service as a “virtual lease line” service in which flows experience low delay low loss service. However, Bennett [BE02] shows that the packet delay of a premium service flow could exceed an arbitrary large delay bound for a network with a large number of nodes under specific conditions. Therefore, the issue of providing “equivalent leased line service” is still an open issue for research.

In other words, the basic idea of flow aggregation in DiffServ is an elegant method that solves the scalability issue when a large number of flows are involved. Priority queuing is then applied to different classes of traffic in this architecture. It provides each flow with a per-hop per aggregate service and its end-to-end characteristics is the convolution of these per-hop per-aggregate behaviors in its path.

Recent work by Hou [HOU01] addresses this problem by applying fair queuing within the premium service. This is an interesting approach but the issue of computational complexity resurfaces due to the sorting requirements of packet finishing time for packet scheduling. The work [SC03] applies Service Curve theory for the service of flows. They obtain interesting theoretical bounds for per-flow service based on service curves. Recently, some researchers have gravitated towards flow aware [ROB2000] networking. In flow aware networking [ROB2000], a flow is defined as a collection of related packet. The idea is to provide guarantee to this “flow” rather than providing the worst-case design for a conventional flow that spans the duration of a connection. By means of packet marking and admission control, a flow aware network could provide QoS to specific flows [K004] by means of service differentiation and rejection of new connections when the network is congested. In this work, packets from streaming flow are marked for service protection. The network gives service precedence to such packets. When part of the network is congested, it would pass congestion notification to the ingress node to reject new request. The scheme protects existing flows by rejecting new flows when the selected path is congested. However, it is not obvious how it can protect against existing flows that may become aggressive and transmit large amount of traffic, causing congestion.

Despite TCP-friendly scenario is often assumed in networks, it does prevent flows from becoming non TCP-friendly or aggressive such as UDP flows which may lead to congestion collapse. To this end, we propose a multiservice framework based on the FDPS algorithm. It provides an “Equivalent Leased Line Service”, addresses the issue of protecting against aggressive real-time flow behaviors, and facilitating efficient resource

utilization in the Next Generation Internet (NGI). To the best of our knowledge, there is little work on practical frameworks for provisioning deterministic QoS guarantee in the literature. We could implement this framework as an enhancement in DiffServ or independently in the Internet paradigm. The rest of the chapter is organized as follows: In Section 6.2, we present the Multiservice Framework with discussions on its building blocks. We provide a summary of the framework and recommendation for future works in Section 6.3.

6.2 Multiservice Framework

In this section, we present the individual building blocks of our proposed multiservice framework. As some of the building blocks share some common functionalities with the existing QoS frameworks, we will focus on the comparisons with existing framework to illustrate the subtle differences.

We have discussed in Section 6.1 that little work is available for a framework with practical implementation of an “Equivalent Leased Line Service (ELLS)”. The “low loss low delay” service in DiffServ, sufficient for many applications such as audio and video transmission, is a “virtual leased line service” but not an ELLS. In this chapter, we propose a new QoS framework. In this framework, we define multiple queuing delay bound guarantees for traffic flows according to their queuing delay budget as defined in Chapter 5. We classify flows during its connection establishment phase and all packets of a flow are considered to belong to the same class during the period of the connection. The purpose of classification of traffic is not only for the sake of scalability (similar to DiffServ) but also for the efficient utilization of the resources of the network. To provide end-to-end protection in this framework, we discuss the essential functioning modules of the framework, including traffic specification for each guaranteed flow, inter-domain service guarantee provisioning, traffic conditioning requirements, our packet scheduling mechanism and packet buffering and discarding for congestion control, etc.

6.2.1 Service Specifications

In today's network environment, we make a pragmatic assumption that deterministic QoS guarantee in a heterogeneous environment is needed. In this environment, there could be many different types of equipment and independent algorithmic implementations. The natural question to ask is: how do we specify a service requirement for a flow? It is worth noting that it is impossible to provide protection to any flow unless we can bind [KN97] its traffic characteristics. In other words, unless we bind the traffic of a flow, it is not meaningful to discuss its QoS guarantee. As a result, we apply the theory of Service Curves [CR91,CR91+,CR95,BOU01,AG99] to provide a simple and quantifiable method that specifies the traffic characteristics and service requirement of a flow.

In Service Curves theory, we define an arrival curve $\bar{S}_i(t)$ and a service curve $S_i(t)$ for flow i . This arrival curve defines the upper bound to the in-profile traffic of the traffic $A_i(t)$ of flow i . Arrival traffic $A_i(t)$ exceeding this bound constitutes out-of-profile traffic which would be treated as part of the best effort traffic in the network. Deterministic QoS guarantee in this framework would be limited to in-profile traffic. In particular, a flow that requires deterministic guarantee must transmit only in-profile traffic. Typically, a regulator at the source shapes the traffic of a flow before the packets are transmitted to the ingress node. By choosing a regulator with envelope $\bar{S}(t)$, only in-profile traffic is injected into the network and is protected. The service curve $S_i(t)$ for flow i defines the minimum service that a flow can receive from the network. In Chapter 2, we could compute the maximum delay and backlog from the arrival curve $\bar{S}_i(t)$ and a service curve $S_i(t)$ for flow i as shown in Figure 2.1. To provide the hard queuing delay bound d^s [CR95, AG99], we can provide a network service curve $S_i(t) = \bar{S}_i(t - d^s)$ for flow i . It ensures that the traffic that arrives at time a receives service not later than time $a + d^s$. This service guaranteed to a flow forms the basis for service requirement for admission control. The network must provide this service at every link along the path of a flow in order to provide the queuing delay guarantee.

Recognizing that the arrival and service curves of flow i provide a concise specification of the traffic envelope and minimum service guarantee for flow i , the framework utilizes these curves to define the envelope of the traffic arrival and minimum service guaranteed to flow i . In other words, to provide deterministic QoS guarantee to flow i with arrival curve $\bar{S}_i(t)$ and delay d^s , the network must provide a service curve $S_i(t) = \bar{S}_i(t - d^s)$. This service curve $S_i(t) = \bar{S}_i(t - d^s)$ for flow i defines the service requirement concisely at each link j along its path. This framework passes this service curve $S_i(t) = \bar{S}_i(t - d^s)$ from one node to another in the path of the flow i as service specification. The detailed discussions on Service Curves can be found in Chapter 2.

In this framework, we apply the idea of class-based treatment of flows. We defined a total of M classes with Class M and 1 representing the highest and lowest priority classes. A queuing delay d_l is associated with class I . For this class-based treatment, we aggregate flows with similar queuing delay budgets into the same class and provide the per-hop per-class treatment. We apply the theory of Service Curves to these aggregate flows by providing different aggregate service curves to provide an efficient deterministic QoS guarantees. As discussed in Chapter 5, the aggregated service curve for class I is given by $\sum_{\forall i} \sum_{\forall i \in F_I} \bar{S}_i(t - d_I^s)$, where F_I is the set of flows in class I and d_I^s is the maximum queuing delay for class I . In this case, the network must provide this minimum aggregate service curve for class I at every node to protect the delay bound of all class I traffic.

6.2.2 Architecture

In this section, we present the architecture for the MultiServ Framework and show how each of the building blocks interact with each other to provide the deterministic QoS guarantee and the approximate Max-Min fairness for the best effort service. In the architecture shown in Figure 6.1, the building blocks in MultiServ include: 1) routing; 2) admission control; 3) flow classification; 3) buffering and policing functions; and 4) a class-based scheduling mechanism.

Figure 6.1 The multiservice framework architecture

The basic idea in this architecture is to keep the core nodes simple and shift the complexity to the bandwidth broker (BB) [QB] and edge nodes where less flows are handled, an idea similar to DiffServ [BL98]. Each domain has one bandwidth broker. The multi-domain architecture would be a concatenation of this basic domain together.

The BB implements most of the functions in the control plane: 1) routing; 2) QoS state database maintenance; 3) admission control; and 4) flow classification. The BB maintains the state database of link and path QoS states [HOU01] in its domain. The path QoS states are derivative of the link states that provide quick assessment of whether a new path could be supported or not. Background process computes the path state information without interfering with its admission process. Having path states in addition to link states would improve the efficiency for path selection in its domain. In the case when a connection involves more than one domain, one BB forwards aggregated service requests to the next BB by means of a service curve specification, until it reaches the BB of the destination domain. Once accepting the connection at the destination domain, its BB would propagate the acceptance to the BB of the source domain. Bandwidth brokers involved would then set the policies at the nodes in the path of the flow. Control is then passed to the data plane at each node. Packet scheduling, buffering and discarding are

main functions in the data plane. We discuss the functions in the control and data plane in the next section.

Let us give an example for the connection establishment phase. When a host signals to its ingress node for a connection in its MultiServ domain, the ingress node signals to the BB for a connection request. The BB would verify its QoS state and path database to determine whether to accept this flow. If it accepts this flow, it would signal to set up the connection and set the flow classification and policing rules for the ingress, egress and core nodes. The ingress node then informs the originating host that the flow is accepted. Control plane passes control over to the data plane for managing the flow of packets in the network. If the connection spans more than one MultiServ domain, the BB of the source domain sends an aggregate RSVP/PATH [DU04] to the BB of the next domain. This BB verifies whether its domain could support this aggregate connection or not. If the responses are positive, it would notify the BB of the originating domain. The source domain BB then sends an acceptance message to the originating host to start data transmission. As the BB would select the source-destination path in this architecture, we will discuss the issue of routing next.

6.2.2.1 Routing

We recall in Chapter 5 that the implementation of routing function in our multiservice framework is subtly different from the current framework of how routing is implemented in the BB of DiffServ domains as follows: routing function is one of the main functions in the BB of each domain in our framework. It provides centralized routing function that allows stateful determination of a path without having errors due to the delayed state update due to propagation and congestion delay. In other words, such delays are small in this centralized system. On the downside, it becomes a critical element in the network that we have to provide a standby BB to protect the system from catastrophic failure. Routing involves selecting the most appropriate path subjected to predefined constraints. Some of these methods include shortest path routing, minimum hop routing, widest path routing and maximum security routing. In this framework, the domain BB selects the path based on the QoS state and path state database maintained

centrally. In this framework, we assume simple standard routing and forwarding protocols such as open shortest path first (OSPF) [IETF94] and the Multiprotocol Label Switching [BA01] in the network. These assumptions do not preclude the possibility of replacing these algorithms for more advanced protocols that utilize the network resources more efficiently in future. One of the key requirements of the routing algorithm is a simple and fast path selection. Based on this path, we would conduct an admission control test and reserve resources along the selected path before admitting a guaranteed flow. As discussed earlier in current section, admission control in the BB verifies that there is sufficient capacity in the path to prevent QoS violation of flows before admitting a flow. We have proposed a class-based packet scheduling mechanism that provides efficient per-flow deterministic QoS guarantee in Chapter 5. In this proposal, we have provided detailed derivation for the admission control conditions. In the next section, we describe the admission control process.

6.2.2.2 Admission Control

Admission Control function is one of the key functions in this architecture. It ensures that there is sufficient capacity in the selected path before admitting a new flow. It verifies that admission of the new flow does cause QoS violation in existing flows. We recall, in section 6.1, that the main purpose of the BB architecture is to add admission control [QB] to the DiffServ architecture [BL98]. For a general network, flows share its finite amount of distributed resources and many of these flows converge at some “hot spots” causing congestion. Though some routing algorithms could spread such traffic load, this complex problem requires information of bandwidth availability in the network. In addition, large variation in number of users during peak and off-peak periods makes network dimensioning a nightmare. Network design based on peak demand is normally not economically viable but dimensioning the network based on average demand could lead to congestion during peak-periods. Therefore, admission control is a necessary function for deterministic QoS guarantee. By means of admission control, DiffServ [BL98] could provide a coarse QoS guarantee for flows using EF per-hop-behavior, a feasible solution for a coarse QoS guarantee in the Internet.

In the MultiServ framework, both routing and admission control are implemented in the BB broker. Each MultiServ domain is controlled by a bandwidth broker (BB). For connecting a number of independent MultiServ domains together, each bandwidth broker controls its own domain. This allows each domain to implement different features and pricing independent of one another. After the routing function in the BB selects the source-destination path, the BB would activate its admission control function to verify that sufficient capacity is available along this path. We have proposed the admission control conditions for admission of a new flow based on class-based deterministic QoS guarantee in Chapter 5. In addition, we have derived theoretical conditions for flow admission. It provides a quantifiable class-to-service mapping for admission control test.

6.2.2.3 Flow Classification

The challenge in flow classification is to find a quantifiable mapping for requirement of a flow to a traffic class. There are many flow classification methods proposed in the literature [WA01,YAS04,ZHE99]. The typical parameters used for classification include type of applications, source and destination addresses and/or port numbers, protocols, applications. The purpose of flow classification is to provide service differentiation among flows according to different QoS requirements. This process categorizes each flow according to their requirements. It provides a mapping from a flow's QoS requirement to a particular traffic class. The scheduling mechanism would provide service differentiation according to the flow classification. For example, VOIP telephony would require an end-to-end delay of less than 100 ms for good quality voice, the question is how we should classify these flows. A natural question to ask is this: Should we classify all voice calls into the same class? With the knowledge that end-to-end delays of different connections could vary significantly from a few milliseconds to about a hundred milliseconds, how should we classify these flows? To the best of our knowledge, there is little work on a quantifiable mapping between these parameters and a class-based service. With little understanding on classification-service relationship in a network, it would be difficult to provide efficient per-flow deterministic QoS guarantee.

In Chapter 5, we proposed a simple flow classification procedure that provides a quantitative mapping for a class-to-service relationship. In this mechanism, we define a total of M QoS classes (configurable) for this network. A different scheduling delay bound is defined for each class of traffic, that is, D_K for class K , where $D_K \neq D_H$ for $K \neq H$. The number of classes and associated delay bound are domain specific and independent of definition in other domains. We classify flows according to their queuing delay budgets. In this method, we classify a flow into the lowest class whose scheduling delay is smaller than its queuing delay budget. Please refer to Chapter 5 for detailed description of the mechanism. After this discussion on flow classification, we present the packet scheduling mechanism that provides service curve of each flow.

6.2.2.4 Packet Scheduling

One of the key objectives of packet scheduling is to service packets in such an order that the QoS requirement of every time sensitive flow is satisfied. Given the chaotic arrivals of packets at a node, the time available for the scheduling decision is normally very small. There has been a plethora of research work on scheduling [BE96, BEN01, VHA03, GOY97, GOY97+, GU04, HA91, II01, KA02, KAT91, LEN04, LI02, LIA03, LIE99, PAR93, PAR94] available in the literature based on fluid flow or packet flow. In Chapter 2, we have provided a review of popular algorithms ranging from fair queuing, weighted fair queuing, weighted round robin to static priority queuing. The requirement for low complexity and high scalability limits the number of practical scheduling algorithms available. Two low computation complexity algorithms today are weighted round robin and priority scheduling. They overcome the scalability issue by flow aggregation into a limited number of classes and providing per class treatment instead of per flow treatment. As the design in DiffServ [BL98] provides service differentiation and coarse QoS guarantee to all users, we require a different algorithm for deterministic guarantee in a multiservice environment.

In our multiservice framework, we implement per-flow deterministic guarantees based on per class treatment of flows. In Chapter 5, we proposed a class-based Flow-state-

dependent Dynamic Priority Scheduling (FDPS) algorithm for this purpose. We show that this mechanism could provide Service Curves assurance. In [SC03], Schmitt has derived the same theoretical result independently. While his [SC03] emphasis is the theoretical service bounds, our focus is to provide an efficient practical algorithm for implementation. In Chapter 3, we show how the FDPS algorithm could provide per flow QoS guarantee based on a specific implementation. In this implementation, we bound the queuing delay of all real-time flows to the small queuing budget among all flow sharing the common link, not an efficient implementation. For efficient service provisioning, we proposed a class-based FDPS mechanism that performs flow classification before applying the FDPS algorithm to each class of traffic scheduled for service. We show that by means of admission control based on Service Curves theory, we could provide an efficient per-flow deterministic QoS guarantee in this mechanism. Please refer to Chapter 3 for detailed discussions on FDPS algorithm.

6.2.2.5 Traffic Policing

As discussed in Section 6.2.1 on service specification, we could not provide QoS guarantee to a flow for unbounded traffic. The purpose of the arrival curve is to define the upper bound of the traffic for service guarantee. For the implementation of the arrival curve for a flow i , we could apply the simple envelope of a token bucket $TB(\sigma_i, \rho_i)$ as a simple policer to separate in-profile and out-of-profile traffic. We discover that simple binary marking [SAH00, SU03] could not provide the protection for the in-profile traffic, the scenario in DiffServ. However, we proposed an enhanced token bucket (ETB) in Chapter 3 for the multiservice framework. Besides identifying in-profile and out-of-profile traffic, the ETB in Chapter 3 marks individual packet according to its flow state for scheduling. For a flow that requires full protection of its traffic, a traffic shaper with the same envelope $TB(\sigma_i, \rho_i)$ would be needed at the source to ensure traffic transmitted is conformant. Please refer to Chapter 3 for more information of the enhanced token bucket.

6.2.2.6 Buffering and Packet Discarding

In this framework, we apply the simple buffering and packet discarding scheme discussed in Chapter 4. It is worth noting that temporary transient overload due to real-time traffic is overcome by buffering. In this case, providing a buffer of appropriate size is the simple solution. The congestion would dissipate very quickly when the arrival rate falls below in link capacity. However, congestion could be due to prolong period of overloading due to out-of-profile traffic or best effort traffic. Recognizing that simple buffering is not a solution due to the rapid rise in backlogged traffic in a severe congestion, we study the behaviors of TCP congestion control and avoidance in Chapter 4. Note that the scheme protects all in-profile packets and discards the non-conformant out-of-profile packets when necessary at congested nodes. We discuss the issue of congestion control in greater details next.

6.2.2.7 Congestion Control

In the Multiservice framework, we address the issue of congestion by dividing the traffic into two parts: 1) in-profile traffic from flows with predefined QoS requirements; and 2) best effort traffic that includes any out-of-profile traffic from time-sensitive flows. We admit in-profile traffic via admission control function with necessary resource reservation along the selected path. Buffer requirements discussed in Chapter 4 ensures that sufficient buffer is available to hold all in-profile traffic. We do not subject unprotected best effort traffic to admission control. In other words, we separate the traffic into two streams consisting of protected in-profile traffic and non-protected out-of-profile traffic. The scheme discards best effort packets when congestion arises.

By scheduling packets using a roughly weighted fair queuing discipline, the service provided is roughly according to the rate of each flow. In [SI05], we have studied the behavior of TCP under the premise of weighted fair queuing and discovered many useful properties. The key advantage of the WFQ discipline is that the channel is roughly Max-Min fairness. In [SI05], we show that the FDPS algorithm provides an implementation for this max-min fairness and inter-works with TCP well. We have shown how the SACC

mechanism in Chapter 4 could work gracefully with TCP together with RED as the router assisted congestion control mechanism. We have shown that the SACC mechanism produces similar performance regardless whether RED is implemented or not, releasing the requirement of choosing appropriate parameter setting in RED. This framework is therefore suitable for the deterministic QoS guarantee and best effort environment.

6.2.2.8 Inter-working with other domains

In last section, we have discussed the issues related to inter-domain connections. In today's environment where different vendors provide different solutions and network implementations, it is a challenging task to provide for QoS guarantee across different autonomous ISP domains in which different technologies are used. Inter-working between different domains with different protocols is one of the main difficulties encountered in connection establishments in early years. This has been resolved by protocol standardization or protocol conversion between different domains. The challenge today is different: How do we provide guarantee service across different domains in a heterogeneous environment? The challenge lies in the different implementations in each domain, a key issue that affects service guarantees.

To communicate the requirement of a flow, an abstraction of service required that could be easily interpreted by every domain is need. Unless we define this abstraction of the QoS guarantees, it is not obvious how deterministic guarantees could be provided for a connection that spans more than one independent domains. As discussed in Section 6.2.1, we advocate the application of Service Curves as a basis for deterministic service guarantees. To provide quantitative guarantee to a flow, we specify the specifications of its arrival curve and service curve derived from its traffic parameters and QoS requirements. These specifications become the abstraction for service guarantees across the domains in its path. The actual implementation of service would be domain dependent yet providing end-to-end guarantees. This abstraction decouples the implementation of one domain from another even if there could be using similar architectures but having different parameter settings due to specific requirements in their domains. For example,

the classification, shaping and policy rules could be different even though two domains have similar architectures. A flow classified in a particular class in one domain could be reclassified into another class in the next domain. In particular, this could be due to insufficient capacity in the same class in the second domain.

To provide inter-domain admission control across heterogeneous domains, an inter-domain architecture that allows independent local policy settings and implementations is needed – requirements that could be met by the Bandwidth Broker [BB]. The bandwidth broker provides an architecture that allows the abstraction of service requirement to pass from one domain to another and the BB of each of these domains would ascertain that sufficient resource is available for the new flow before making confirmation or making further request to another domain. The basic operation is similar to the BB architecture for the DS domain except for the specification of service requirement based on Service Curve. The abstraction based on service curves clearly defines the service requirement without restrictions on the actual implementations. Service Curves for inter-domain admission control has some very useful properties in a heterogeneous environment. The requirement for an aggregated flow [MA97] is sent to the next BB in terms of aggregated arrival curve and aggregated service curve. This framework allows the co-existence of different technologies and implementation, suitable for connection between independent domains.

6.3 Summary and Future Work

We have proposed a new QoS provisioning framework, called Multiservice Framework (MultiServ). MultiServ is a very efficient QoS framework which is based a novel packet Scheduling algorithms FDPS. It possesses many good features. Table 6.1 shows a comparison among IntServ, DiffServ and MultiServ, in terms of various aspects. Note that queuing delay is the parameter in traffic classification in the Multiservice Framework.

This framework utilizes the bandwidth broker for admission control and allows inter-domain reservation. This leads to high scalability by means of reservation aggregation

before sending to the BB of the next domain. Applying the theory of Service Curves, we achieve a per-class scheduling delay bound formulated a simple admission control test (details described in Chapter 5). Besides providing a scalable framework for deterministic QoS guarantee in a multiservice environment, the framework shares many desired properties of IntServ and DiffServ as shown in Table 6.1.

In summary, we have presented a multiservice framework that provides deterministic QoS guarantee with different queuing delay bounds together with best effort service in a multiservice environment. While IntServ addresses the issue of deterministic QoS, its lack of scalability leads to the development of a scalable DiffServ architecture but it provides coarse QoS guarantees by means of per-aggregate per-hop-behavior service.

Although statistic guarantee is unavailable in this framework, it is worth noting that the current focus of our work is providing a range of deterministic QoS guarantees in the NGL. There is sufficient flexibility in this framework that statistic guarantee could be included, a possible area for future work. For example, we could allocate the priority class just above the best effort class to provide statistic QoS guarantees. Measurement based admission control could be added to the framework for this stochastic QoS guarantee.

Table 6.1 COMPARISON OF INTSERV, DIFFSERV AND MULTISERV FRAMEWORK

Criteria	IntServ	DiffServ	MultiServ
Granularity of service differentiation	Individual flow	Aggregate of flows	Aggregated flow
State in routers (e.g., scheduling, buffer management)	Per flow	Per aggregate	Per aggregate
Traffic classification basis	Several header fields	DS field	Queuing delay budget
Type of service	Deterministic guarantees	Relative assurance	Deterministic guarantees
Admission control	Required	Required for coarse QoS guarantee	Required
Signaling protocol	Required (RSVP)	Not required for relative schemes	Required (BB)
Coordination for service	End-to-end	Local (per-hop)	End-to-end
Scope of service	A unicast or multicast path	On per-hop basis at each node	A unicast or multicast path
Scalability	Limited by the number of flows	Limited by the number of classes	Limited by the number of classes
Network accounting	Based on flow characteristics and QoS requirements	Based on class usage	Based on rate and delay class usage
Network management	Similar to circuit switching networks	Similar to existing IP networks	Similar to existing IP network
Inter-domain deployment	Multilateral agreements	Bilateral agreements	Bilateral agreement

CHAPTER 7

CONCLUSION AND RECOMMENDATION FOR FUTURE WORK

7.1 Summary and Contribution

The research of this PhD project is motivated by the identification of the need of efficient provisioning of deterministic QoS guarantee in the Next Generation Internet (NGI). Though a group of the Internet community thinks that a coarse QoS that could be provided by Differentiated Service is sufficient, we have different opinion. We argued that this view would have neglected an important group of users who are using expensive leased line due to the high financial value of their transactions. Due to the fiscal value of their transaction, they are willing to pay for leased line QoS. Without providing deterministic QoS guarantees, it is almost impossible to attract this important group of users to migrate to the NGI and provide an integrated service in NGI. This led to the researches on packet scheduling algorithm and related subjects that associate with QoS guarantee provisioning.

In this thesis, we have provided a review of the major research work in relevant areas, including scheduling, service curves, admission control, flow and packet classifications, congestion control together with an over view of the known QoS provisioning architectures: Integrated Service and Differentiated Service architectures. The search for a practical algorithm leads to our proposal of the Flow-state-dependent Dynamic Priority Scheduling (FDPS) algorithm that schedules the packets according to the state of individual flows. We employ a novel enhanced token bucket to extract the current state of a flow to assign a priority to its packets. Packets from different flows would then compete for service based on their priority. We applied the theory of Service Curves to quantify the performance of this algorithm and derive the mapping between the service curve parameters and implementation parameters. Admission control is added to ensure that

sufficient capacity is available for a new flow before admission. Based on an analysis by means of service curves, we showed that the FDPS algorithm could provide service curves assurance for per-flow QoS guarantees. The novel mechanism of the FDPS algorithm schedules packets according to the state of its flow and packets that belong to different flows would compete for service based on the priority assigned by this algorithm. This property is shown to provide per flow QoS guarantee but with a snag – all flows are provided the same low queuing delay. A similar property is identified by Stoica [ST96] for the case of DiffServ for the premium service. On the downside, our practical solution based on FDPS algorithm leads to a short queuing delay for all deterministic flows. If we could spread this queuing delay, we could have a more efficient utilization of the network.

It is worth noting that FDPS identifies in-profile and out-of-profile traffic and assigns a priority to each packet. Out-of-profile packets and best effort packets are assigned negative priority. The scalar value of such a packet would depend on the measure of non-conformance. We apply this property for the scheduling of best effort packets.

In Chapter 4, we investigated the performance of the FDPS algorithm for best effort service environment. We have shown that this algorithm would provide a rough max-min fair share allocation of bandwidth to best effort flows, a desired property for best effort traffic. We showed that the algorithm removes the throughput dependence of TCP flow on RTT. In particular, our simulations show that this algorithm can prevent congestion collapse in the presence of aggressive non-TCP friendly flows, as the FDPS algorithm discards more packets from these aggressive flows leading to their failure. The idea of getting more bandwidth during the congestion period by transmitting more traffic is not rewarded by this algorithm. By means of simulation, we showed that the FDPS algorithm works well with Active Queue Management scheme such as RED. The simulation results also show that there is little difference in performance of TCP flows with and without RED. Hence, the difficulty of setting the minimum and maximum thresholds for RED does not apply for a network using FDPS scheduling.

To address the issue of efficient application of the FDPS algorithm, we have applied the idea to the flow aggregation in the latest work in Chapter 5. There is little work that provides a quantitative mapping from classification parameters to service requirement. Rather than passing responsibility to the users to select the service required, a simple mapping is needed. To this end, we apply the theory of service curves to formulate a mapping from user requirement to service requirement. We proposed a new flow classification scheme based on queuing delay budget. Besides the maximum delay for a flow and its fixed overheads in the delivery of traffic in a path, we could calculate the allowable queuing delay, which is termed “queuing delay budget”, without violating its QoS requirement. After classifying a flow into a particular queuing delay class, we apply class-based scheduling to its traffic followed by dynamic scheduling using the FDPS algorithm within each class. This simple scheduling scheme provides per-flow protection based on class-based treatment of flows. By means of Service Curves theory, we show that this mechanism could provide per-flow deterministic QoS guarantee. The results are also verified by computer simulation using NS-2.

After discovering the useful features of the FDPS algorithm for congestion control and per-flow deterministic QoS guarantee with class based treatment of flow, we proposed a Multiservice Framework for the Next Generation Internet (NGI) in Chapter 6. In this work, we provide a very brief review of the two key existing architectures: IntServ and DiffServ. Many desirable features are already available in these architectures but we realize that deterministic QoS guarantee discussed in Chapter 5 and congestion control in Chapter 4 would be desirable enhancements. These enhancements would allow the NGI to support all existing traffic, making the NGI a true integrated service network. We have divided the functions in this framework into the control plane and data plane. All control plane functions are implemented in a Bandwidth Broker. In this framework, the BB maintains the QoS state and path state in its databases, allowing stateful admission of flows while keeping the router nodes simple. The control plane functions include Routing, Flow classification, Signaling and Admission control. Some or all of these functions are activated in connection establishment and termination phases. Once a flow is admitted, control is passed to the data plane. The functions in the data plane include

packet forwarding, traffic policing, packet marking and scheduling, and congestion control. In this framework, our design would utilize existing algorithms and protocols as much as possible to minimize “reinventing of the wheel” syndrome. Hence, existing routing and forwarding protocols such as OSPF and MPLS form part of the framework. Specifically, we mentioned that this framework could use more advanced routing and forwarding protocols when they become available. To establish a new connection, the routing and admission control functions in the BB would select the route and verify that sufficient capacity is available in the selected path. Once a flow can be admitted, the BB would signal to the nodes to update forwarding tables and traffic policies at each node in the path of the flow. One of the key features of this framework is the abstraction of service requirement of a flow by means of a service curve derived from its traffic characteristics and delay requirement. By means of this service abstraction, the aggregate inter-domain service requirement could be requested. However, the responsibility of providing this service is left to each domain without any restriction on the technology involved. At the heart of this framework is the packet scheduling based on class-based treatment of flows and using the FDPS algorithm for scheduling within a class of traffic. The protection provided to a guaranteed flow makes this a desirable mechanism.

7.2 Recommendation for Future Work

In the multiservice frame, we mentioned that statistic guarantee could be provided in class 2 – the better than best effort class of traffic. Further studies are needed to quantify the performance of this low priority statistic guarantee. There is also a need to understand its impact on best effort traffic and network efficiency.

We have not considered pricing in this framework. With equivalent leased line quality provided, we could maximize a utility function based on revenue or number of admitted connections. Internet Service providers would be very interested in this pricing model. One may apply game theory to emulate different scenarios and computer simulations could test assumptions that could not be analyzed mathematically.

The interest in video multicasting in the media industry is high nowadays especially with interest of the ISPs. Quality of video and cost of provisioning are two of the key issues in the area. If the network is allow to carry too much traffic, quality would be seriously affect when congestion arise. Too much over provisioning leads to high cost in provisioning. Though we have studied the performance of unicast flow in a congested FDPS network, the work could be extended to a multicast flow in which a source is transmitting to a large number of users. Quality of video reception and efficiency could be two of the key issues for analysis.

Implementation of this the FDPS algorithm could be the next phase of research. Some issues which are not considered in the study would surface in the implementation stage. An engineering solution would then be required. We will quote “the test of the car is when the tires hit the road”.

Author's Related Publications

1. Chee-Kheong Siew, Gang Feng, Fei Long, and Meng-Hwa Er, "Congestion Control Based on Flow-state-dependent Dynamic Priority Scheduling," IEE Proc. Communications, vol. 152, no. 5, Oct. 2005, pp. 548-558.
2. C. K. Siew & M. H. Er, "A New Multiservice Provisioning Mechanism with Service Curves Assurance for Per-class Scheduling Delay Guarantees" IEE Proceedings - Communications, vol. 153, no. 6, December 2006, p. 846-855.
3. Chee-Kheong Siew, Gang Feng, Fei Long, and Meng-Hwa Er, "A flow-sate-dependent Dynamic Priority Scheduling Algorithm for Service Curves Assurance," submitted to journal of Communications and Networks.
4. Chee-Kheong Siew and Sheng Jin, "Admission Control based on a Simple Core Network with Intelligent Ingress Node," ICISP 2003, vol. 2, Dec. 2003, pp. 1268-1272.
5. Chee-Kheong Siew, Gang Wu, Gang Feng, "On-demand QoS multipath routing," ICCS 2002, vol. 1, Nov. 2002, pp. 589-595.
6. C.-K. Siew, J.Tang, G.-B. Huang, J. Sheng and Meng-Hwa Er, "Dynamic Priority Scheduling Discipline," IASTED Conf., CCN 2002, pp. 110-115..
7. Xun Yang, Gang Feng, Chee Kheong Siew, "Call Admission Control for Multiservice Wireless Networks with Bandwidth Asymmetry between Uplink and Downlink," IEEE Trans. Vehicular Technologies, Vol. 55, no. 1, Jan. 2006, pp. 360 – 368.
8. Fei Long, Gang Feng, Chee-Kheong Siew, "Channel States Dependent Fair Service: A New Packet Scheduling Algorithm for CDMA," Computer Networks, vol. 49, no 2, Oct. 2005, pp. 201-206.
9. Gang Feng, A. K. Agarwal, A. Jayaraman, and Chee-Kheong Siew, "A Modified RED gateway Algorithm for Congestion Avoidance under Bursty Traffic," IEEE Communications Letters, vol. 8, no. 5, May 2004, pp 323-325.

REFERENCES

- [AG99] R. Agrawal, R.L. Cruz, C. Okino, and R. Rajan. Performance bounds for flow control protocols. *IEEE/ACM Transactions on Networking*, 7(3):310-323, June 1999
- [AG98] R. Agrawal, R. Cruz, C. Okino, and R. Rajan, "A framework for adaptive service guarantees," in *Proc. Allerton Conf. on Comm. and Comp.*, (Monticello, IL), Sept 1998.
- [AL04] C. Albuquerque, B. J. Vickers and T. Suda, "Network Border Patrol: Preventing Congestion Collapse and Promoting Fairness in the Internet," *IEEE/ACM Transactions on Networking*, Vol.12, No.1, Feb. 2004, pp173-186
- [AR02] Arnaud Legout and Ernst W. Biersack, "Revisiting the Fair Queuing Paradigm for End-to-End Congestion Control," *IEEE Network*, September/October 2002, pp 38-46
- [ATH01] S. Athuraliya, S. H. Low, V. H. Li, and Q. Yin, "REM: Active queue management," *IEEE Networks*, 2001, Vol 15, pp. 48-53.
- [BA01] A. Banerjee, J. Drake, J. P. Lang and B. Turner, K. Kompella and Y. Rekhter, "Generalized Multiprotocol Label Switching: An Overview of Routing and Management Enhancements," *IEEE Communications Magazine*, Vol. 39, January 2001, pp. 144-150
- [BE96] J. Bennett and H. Zhang, "WF²Q: Worst case weighted fair queueing," *IEEE INFOCOM*, pp. 120-128, Mar 1996.
- [BE02] J. C. R. Bennett, K. Benson, A. Charny, W.F. Courtney, J.Y. Le Boudec, "Delay jitter bounds and packet scale rate guarantee for expedited forwarding," *IEEE/ACM Transactions on Networking*, Volume: 10 , Issue: 4 , Aug. 2002 Pages:529 - 54
- [BEN01] B. Bensaou, D. Tsang, and K.T.Chan, "Credit-based fair queuing (CBFQ): a simple service-scheduling algorithm for packet-switched networks," *IEEE/ACM Transactions on Networking*, vol. 9, no. 5, pp. 591-604, 2001
- [BER98] A. W. Berger and W. Whitt, "Effective Bandwidths with Priorities," *IEEE/ACM Transactions on Networking*, Vol. 6, no. 4. August 1998, pp. 447-460
- [BL98] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, "An Architecture for Differentiated Services," RFC 2475, December 1998. Available <ftp://ftp.isi.edu/in-notes/rfc2475.txt>
- [BO03] T. Bonald, J. Roberts, "Congestion at flow level and the impact of user behaviour," *Computer Networks* Vol 42, pp 521-536, 2003

- [BOU01] J-Y Le Boudec, P. Thiran, Network calculus, Springer-Verlag, ISBN 3-540-42184-X, June 2001.
- [BOU96] J-Y Le Boudec, "Network calculus made easy," Technical Report 96/218, DI-EPFL, CH-1015 Lausanne, Switzerland, December 1996
- [BR94] R. Braden, D. Clark and S. Shenker, "Integrated Services in the Internet Architecture: an Overview," RFC 1633. June 1994. Available <ftp://ftp.isi.edu/in-notes/rfc1633.txt>
- [BR98] B. Braden et al., Recommendations on queue management and congestion avoidance in the Internet, RFC 2309, Internet Engineering Task Force (1998).
- [BRA95] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End-to-end congestion control on a global Internet" IEEE Journal on Selected Areas in Communications, Vol. 13, No. 8, 1995
- [CE01]] C. Cetinkaya, V. Kanodia and E.W. Knightly, "Scalable Services via Egress Admission Control," IEEE Transactions on Multimedia, Vol. 3, no.1, March 2001, pp 69-81
- [CH94] C-S Chang, "Stability, Queue Length, and Delay of Deterministic and Stochastic Queueing Networks," IEEE Transactions on Automatic Control, Vol. 39, No. 5, May 1994
- [CHA03] H.M. Chaskar and U. Madhow, "Fair Scheduling with tunable latency: a round-robin approach," IEEE/ACM Transactions on Networking, Vol. 11, Aug. 2003, pp. 592-601
- [CHI89] Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks", Computer Networks and ISDN Systems 17, 1989, pp 1-14
- [CHU00] C.-N. Chuah, L. Subramanian, R. H. Katz and A. D. Joseph, "QoS Provisioning Using A Clearing House Architecture," IWQOS 2000, pp. 115-124
- [CR91] R. L. Cruz, "A calculus for network delay. I: network elements in isolation," IEEE Transactions on Information Theory, Vol.37, No.1, Jan 1991, pp.114-131
- [CR91+] R. L. Cruz, "A calculus for network delay. II: network analysis," IEEE Transactions on Information Theory, Vol.37, No.1, Jan 1991, pp.132-14
- [CR95] R. L. Cruz, "Quality of service guarantees in virtual circuit switched networks," IEEE Journal on Selected Areas in Communications, Vol.13, No.6, Aug.1995, pp.1048-1056
- [CR98] R.L. Cruz, "SCED+: efficient management of quality of service guarantees," IEEE Computer and Communications, Infocom'98, Vol. 2, 29 March-2 April 1998, pp. 625-634

- [DU04] Z.H. Duan, Z. L. Zhang, Y.T. Hou and L.X Gao, "A core stateless bandwidth broker architecture for scalable support of guaranteed services," IEEE Trans. on Parallel and Distributed Systems, Vol. 15, 2 Feb 2004, pp. 167-182
- [EI93] A. I. Elwalid and D. Mitra, "Effective Bandwidth of General Markovian Traffic Sources and Admission Control of High Speed Networks," IEEE/ACM Transactions on Networking, Vol. 1, no. 3, June 1993, pp. 329-343
- [EU03] D.Y. Eun and N.B Shroff, "A Measurement-Analytic Approach for QoS Estimation in a Network Based on the Dominant Time Scale," IEEE/ACM Transactions on Networking, Vol. 11, no. 2, April 2003, pp. 222-235
- [FE02] W. C. Feng, Kang G. Shin, D. K. Kandlur and D. Saha, "The Blue Active Queue Management Algorithm", IEEE/ACM Transactions on Networking Vol. 10. NO. 4. August 2002.
- [FI98] V. Firoiu, J. Kurose, D. Towsley, "Efficient admission control of piecewise linear traffic envelopes at EDF schedulers", IEEE Transactions on Networking, Vol 6, Oct 1998, pp. 558-570
- [FI02] V. Firoiu, J.-Y. L. Boudec and D. Towsley, "Theories and Models for Internet Quality of Service," Proceedings of the IEEE, Vol. 90, no. 9, September 2002, pp. 1565-1591
- [FL93] S. Floyd and V. Jacobson, "Random early detection gateways for congestion avoidance," IEEE/ACM Transactions on Networking, Aug 1993, pp 397-423
- [FL95] Sally Floyd and Van Jacobson, "Link-sharing and resource management models for packet networks", IEEE/ACM Transactions on Networking, 3(4), August 1995.
- [FL00] S. Floyd, J.Pedhye and J.Widmer, "Equation-based congestion control for unicast applications", Proc. of ACM SIGCOMM, 2000
- [GE97] L. Georgiadis, R. Guérin, and A. Parekh, "Optimal multiplexing on a single link: Delay and buffer requirements," IEEE Trans. Infor. Theory, vol. 43, pp. 1518–1535, Sep 1997.
- [GE96] L. Georgiadis, R. Guérin, V. Peris, and R. Rajan, "Efficient support of delay and rate guarantees in an Internet," in Proc. SIGCOMM, pp. 106–116, Aug 1996
- [GE96+] L. Georgiadis, R. Guérin, V. Peris, and K. Sivarajan, "Efficient network QoS provisioning based on per node shaping, IEEE trans. On Networking, Vol 4, Aug 1996, pp. 482-501
- [GO94] S. J. Golestani, "A self-clocked fair queueing scheme for broadband applications," in Proceedings of IEEE INFOCOM 94, (Toronto, CA), pp. 636–646, June 1994

- [GOY97] P. Goyal, H. M. Vin, and H. Cheng, "Start-time fair queuing: a scheduling algorithm for integrated services packet switching networks," *IEEE/ACM Transactions on Networking*, vol. 5, no. 5, pp. 690-704, 1997.
- [GOY97+] P. Goyal and H. M. Vin, "Generalized guaranteed rate scheduling algorithms: A framework," *IEEE/ACM Transactions on Networking*, vol. 5, no. 4, pp. 561-571, 1997
- [GU04] C.X. Guo, "SRR: an $O(1)$ time-complexity packet scheduler for flows in multiservice packet networks," *IEEE/ACM Transactions on Networking*, Vol. 12, Dec. 2004, pp. 1144-1155
- [HA91] E.L. Hahne, "Round-robin scheduling for max-min fairness in data networks," *IEEE Journal on Selected Areas in Communications*, Vol. 9, Sep 1991, pp. 1024-1039
- [HAO99] Hao Che and San-qi Li, "MPOA Flow Classification Design And Analysis," *Infocom,99*, pp 1497-1504.
- [HO01] D. Hong, C. Albuquerque, C. Oliveira, and T. Suda, "Evaluating the impact of emerging streaming media applications over TCP/IP performance," *IEEE Commun. Mag.*, vol.39, no.4, pp76-82, Apr, 2001
- [HOU01] Yiwei T. Hou, Zhenhai Duan, Zhi-Li Zhang, and Takafumi CHUJ, "Providing Scalable Support for Multiple QoS Guarantees: Architecture and Mechanisms. *IEICE Transaction on Communications*, E84-B (10), October. 2001. pp2830-2849
- [HU93] T.-Y. Huang and J.-L. C, "Performance analysis of a dynamic priority scheduling method in ATM networks," *IEE Proceedings I Communications, Speech and Vision*, Vol. 140, Aug 1993, pp. 285-290
- [II01] K. Iida, T. Takine, H. Sunahara and Y. Oie, "Delay Analysis for CBR Traffic Under Static-Priority Scheduling," *IEEE/ACM Transactions on Networking*, Vol. 9, no. 2, April 2001, pp. 177-185
- [IETF94] Internet Engineering Task Force, Open Shortest Path First Protocol (OSPF) Version 2, Technical Report, RFC 1583, Network Working Group, 1994
- [IETF99] An expedited forwarding PHB, V. Jacobson, K. Nichols, K. Poduri, RFC 2598, IETF proposed standard, June 1999
- [IETF98] "Recommendations on Queue Management and Congestion Avoidance in the Internet", Internet RFC 2309, April, 1998.
- [IETF01] "The Addition of Explicit Congestion Notification (ECN) to IP", Internet RFC 3168, September, 2001
- [IETF97] "Resource ReSerVation Protocol (RSVP) -- Version 1 Functional Specification", Internet RFC 2205, September, 1997
- [JA88] V.Jacobson "Congestion avoidance and control", In *Proceedings of SIGCOMM'88*, Stanford CA, Aug. 1988, ACM

- [JA90] V. Jacobson, "Modified TCP congestion avoidance algorithm," Technical Report, April 1990
- [KA02] S.S. Kanhere, H. Sethu and A.B. Parekh, "Fair and efficient packet scheduling using Elastic Round Robin," IEEE Transactions on Parallel and Distributed Systems, Vol. 13, Mar 2002, pp.324-336
- [KAT91] M. Katevenis, S. Sidiropoulos and C. Courcoubetis, "Weighted round-robin cell multiplexing in a general-purpose ATM switch chip," IEEE Journal on Selected Areas in Communications, Vol. 9, Oct. 1991, pp. 1265-1279
- [KATS97] Y. Katsube, K. Nagami, S. Matsuzawa and H. Esaki, "Internetworking Based on Cell Switch Router – Architecture and Protocol Overview," Proceedings of the IEEE, Vol. 85, no. 12, December 1997, pp. 1998-2006
- [KI01] H.S. Kim and N.B. Shroff, "Loss Probability Calculations and Asymptotic Analysis for Finite Buffer Multiplexers," IEEE/ACM Transactions on Networking, Vol. 9, no. 6, December 2001, pp. 755–768
- [KN95] E. Knightly, D. Wrege, J. Liebeherr, and H. Zhang, "Fundamental Limits and Trade-offs of Providing Deterministic Guarantees to VBR Video Traffic ," Proc. of ACM SIGMETRICS'95, 1995
- [KN99] E. Knightly, "Resource allocation for multimedia traffic flows using rate-variance envelopes," ACM Multimedia Systems Journal, vol. 7, no. 6, pp. 477-485, 1999
- [KN97] E. Knightly and H. Zhang, "D-BIND: An Accurate Traffic Model for Providing QoS Guarantees to VBR Traffic," IEEE/ACM Transactions on Networking, 5(2): 219-131, April 1997
- [KN98] E. Knightly, "Enforceable Quality of Service Guarantees for Bursty Traffic Streams," in Proceedings of IEEE INFOCOM '98, San Francisco, CA, March 1998, pp. 1-8
- [KN97+] E. Knightly and N.B. Shroff, "Admission Control for statistical QoS: Theory and Practice," IEEE Network Magazine, Vol. 13, April 1997, pp. 20-29
- [KO04] A. Kortebi, S. Oueslati, J. Roberts, "Cross-protect: implicit service differentiation and admission control", in Proceedings of Workshop on High Performance Switching and Routing(HPSR), 2004. pp. 56-60
- [KU01] S. Kunniyur and R. Srikant, " Analysis and design of an adaptive virtual queue (AVQ) algorithm for active queue management." Proceeding of ACM Sigcomm, San Diego, USA, 2001, pp.123-134.
- [LE02] A. Legout and E. W. Biersack, "Revisiting the Fair Queuing Paradigm for End-to-End Congestion Control," IEEE Network, September/October 2002, pp 38-46
- [LEN04] L. Lenzini, E. Mingozzi and G. Stea, "Tradeoffs between low complexity, low latency, and fairness with deficit round-robin scheduling," IEEE/ACM Transactions on Networking, Vol. 12, Aug. 2004, pp. 681-693

- [LI02] C. Z. Li and E. W. Knightly, "Coordinated Multihop Scheduling: A Framework for End-to-End Services," IEEE/ACM Transactions on Networkin, December 2002, pp. 1-13
- [LIA03] J. Liang, T.N. Arvanitis and S.I. Woolley, "Fair weighted round robin scheduling scheme for DiffServ networks," Electronics Letters, Vol. 39, Feb 2003, pp. 333-335
- [LIE96] J. Liebeherr, D. Wrege, and D. Ferrari, "Exact admission control for networks with a bounded delay service," IEEE/ACM Transactions on Networking, vol. 4, no. 6, pp. 885-901, 1996
- [LIE99] J. Liebeherr and D. E. Wrege, "Priority Queue Schedulers with Approximate Sorting in Output-Buffered Switches," IEEE Journal on Selected Areas in Communications, Vol 17, June 1999, pp.1127-1144
- [LIN97] S. Lin and N. McKeown, "A Simulation Study of IP Switching," Tech. report CSL-TR-97-720, April 97
- [LIU00] H. Liu and P. Mouchtaris, "Voice over IP Signaling: H.323 and Beyond," IEEE Communications Magazine, Vol. 38, October 2000, pp.142-148
- [LO02] S. H. Low and F. Paganini, "Internet congestion control", IEEE Control Systems Magazine, Feb 2002
- [MA97] Mankin, A., Baker, F., Braden, B., Bradner, S., O'Dell, M., Romanow, A., Weinrib, Resource Reservation Protocol (RSVP) Version 1 Applicability Statement, RFC 2208, September 1997
- [MAN04] H.A. Mantar, J.S. Hwang, I.T. Okumus and S.J. Chapin, "A scalable model for interbandwidth broker resource reservation and provisioning," IEEE Journal on Selective Areas in Communications, Vol. 22, 10 Dec 2004, pp. 2019-2034
- [MC96] McCanne, S., Jacobson, V., and M. Vetterli, "Receiver- driven Layered Multicast," ACM SIGCOMM, 1996.
- [MI99] D. Mitra and K.G.Ramakrishnan, "A Case Study of Multiservice, Multipriority Traffic Engineering Design for Data Networks," Proc. IEEE GLOBECOM 99, pp.1077-1083, Dec., 1999
- [MIT03] D.Mitra and Q.Wang, "Stochastic Traffic Engineering, with Applications to Network Revenue Management," Proc. IEEE INFOCOM 2003, San Francisco, April 2003
- [MO00] J. Mo and Jean Walrand, " Fair end-to-end window-based congestion control", IEEE/ACM Transactions on Networking, Vol. 8, No. 5, Oct 2000, pp 556-567
- [MO92] A.R. Modarressi and R.A. Skoog, "An Overview of Signaling System No. 7," proceedings of the IEEE, Vol. 80, no. 4, April 1992, pp. 590-606
- [MY03] E. Mykoniati, C. Charalampous, P. Georgatsos, T. Damlatis, D. Goderis, P. Trimintzios, G. Pavlou, D. Grffin, "Admission control for providing QoS in

- DiffServ IP Networks: The Tequila Approach”, IEEE Communications Magazine, Vol. 41, Jan 2003, pp. 38-44
- [NS] Network simulator NS-2. Available at [Http://www.isi.edu/nsnam/ns/](http://www.isi.edu/nsnam/ns/).
- [NI99] K. Nichols, V. Jacobson, and L. Zhang, “A two-bit differentiated services architecture for the Internet,” RFC 2638 Internet Engineering Task Force, July 1999.
- [PA00] R. Pan, B. Prabhakar, and K. Psounis, “CHOKe - A stateless queue management scheme for approximating fair bandwidth allocation,” INFOCOM, March, 2000, pp. 942-951.
- [PAN01] Panos Gevros, Jon Crowcroft, Peter Kirstein, and Saleem Bhatti, “Congestion Control Mechanisms and the Best Effort Service Model,” IEEE Network, May/June 2001, pp. 16-26
- [PAD99] J. Padhye, J. Kurose, D. Towsley and R. Koodli, “A model based TCP friendly control protocol”, Proc. of IEEE NOSSDAV, June 1999
- [PAR93] A. K. Parekh and R. G. Gallager, “A generalized processor sharing approach to flow control in integrated services networks: the single-node case,” IEEE/ACM Transactions on Networking, vol. 1, no. 3, pp. 344-357, 1993
- [PAR94] A. K. Parekh and R. G. Gallager, “A generalized processor sharing approach to flow control in integrated services networks: the multiple node case,” IEEE/ACM Transactions on Networking, Vol. 2, No. 2, April 1994, pp. 137-150
- [QB] QBONE Bandwidth Broker Architecture. Available at <http://ing.ctit.utwente.nl/WU3/QBoneWWW/BBroker/>
- [PY03] K. Pyun, “Delay-Optimal Service Curve Allocation Scheme,” IEEE Communications Letters, Vol. 7, no. 10, October 2003, pp. 505-507
- [RO00] J.W. Roberts and S. Oueslati Boulahia, “Quality of Service by Flow Aware Networking,” Philosophical Transactions of The Royal Society of London, series A vol. 358, number 1773, August 2000
- [RO04] J. Roberts, “A survey on statistical bandwidth sharing,” Computer networks, vol 45, pp 319-332, 2004
- [RO04+] J. Roberts, “Internet Traffic, QoS and Pricing,” Proceedings of the IEEE, Volume: 92, Issue: 9, Sept. 2004, Pages: 1389 – 1399
- [ROB96] Robert Hinden, “IP Next Generation Overview”, Communications of the ACM, Volume 39, Issue 6 (June 1996) Pages: 61 - 71
- [SA97] J. Sagih, P.B. Danzig, S.J. Shenker, L. Zhang, “A measurement base-based admission control algorithm for Integrated service packet network”, Transactions on Networking, Vol 5, Feb 1997, pp. 56-70

- [SAH00] S. Sahu, P. Nain, C. Diot, V. Firoiu, D. Towsley, "On achievable service differentiation with token bucket marking for TCP. Sigmetrics 2000, pp.23-33
- [SAH00+] S. Sahu, V. Firoiu, D. Towsley, and J. Kurose, "Traffic models and admission control for variable bit rate continuous media transmission with deterministic service", Performance Evaluation 40, 2000, pp 115-133
- [SAR99] H. Sariowan, R.L. Cruz and G.C. Polyzos, "SCED: a generalized scheduling policy for guaranteeing quality-of-service," IEEE/ACM Transactions on Networking, Vol. 7, Oct 1999, pp. 669-684
- [SAR95] H. Sariowan, R.L Cruz and G.C. Polyzos, "Scheduling for quality of services guarantees via service curves," Proceedings Computer Communications and Networks, Sept. 1995, pp. 512-520
- [SAT02] K-I. Sato, N. Yamanaka, Y. Takigawa, M. Koga, S. Okamoto, K. Shiimoto, E-I. Oki and W. Imajuku, "GMPLS-Based Photonic Multilayer Router (Hikari Router) Architecture: An Overview of Traffic Engineering and Signaling Technology," IEEE Communications Magazine, Vol. 40, March 2002, pp. 96-101
- [SC03] J. Schmitt, P. Hurley, M. Hollick, and R. Steinmetz, "Per-flow guarantees under class-based priority queueing", GLOBECOM '03,
- [SH95] S. Shenker, "Fundamental Design Issues for the Future Internet," IEEE Journal on Selected Areas in Communications, Vol. 13, no. 7, September 1995, pp. 1176-1188
- [SHE98] T. Shepard and C. Partridge, "When TCP Starts Up With Four Packets Into Only Three Buffers ", RFC 2416, Sept 1998.
- [SHR96] M. Shreedhar and G. Varghese, "Efficient fair queuing using deficit round-robin," IEEE/ACM Transactions on Networking, Vol. 4, Jun 1996, pp. 375-385
- [SHW94] A. Shwartz and A. Weiss, "*Large Deviations for Performance Analysis*", Chap.& H, ISBN 0-412-06311-S, 1994
- [SI02] C. K Siew, J.Tang, G.B.Huang, J. Sheng, and M.H. Er, "Dynamic Priority Scheduling discipline" IASTED Communications and Computer Networks (CCN 2002), Cambridge, USA, Nov 2002
- [SI05] Chee-Kheong Siew, Gang Feng, Fei Long, and Meng-Hwa Er, "Congestion Control Based on Flow-state-dependent Dynamic Priority Scheduling," to appear in IEE Proceedings - Communications
- [ST96] D. Stiliadis, "Traffic scheduling in Packet-switched networks: Analysis, Design and Implementation", PhD Dissertation, University of California, Santa Cruz, 1996
- [ST98] D. Stiliadis and A. Verma, "Rate-proportional servers: A design methodology for fair queuing algorithms," IEEE/ACM Transactions on Networking, vol. 6, no. 2, pp. 164-174, 1998

- [STO00] I. Stoica, "Stateless core: A scalable approach for quality of service in the internet," PhD Dissertation, CMU, EEE, 2000
- [STO98] I. Stoica, S. Shenker, H. Zhang, "Core-Stateless Fair Queuing: Achieving Approximately Fair Bandwidth Allocations in High Speed Networks," *Computer Communication Review*. vol.28, no.4, p. 118-30, October 1998
- [SU03] H. Su and M. Atiquzzaman, "Comprehensive performance model of differentiated service with token bucket marker," *IEE Proc.-Commun.*, Vol. 150, no. 5, October 2003, pp. 347-353
- [SUR97] S. Suri, G. Varghese, and G. Chandranmenon, "Leap forward virtual clock: a new fair queuing scheme with guaranteed delays and throughput fairness," in *Proc. IEEE INFOCOM 97*, (Kobe, Japan), pp. 557-565, Apr 1997.
- [TE99] B. Teitelbaum, S. Hares, L. Dunn, R. Neilson, V. Narayan and F. Reichmeyer, "Internet2 QBone: building a testbed for differentiated services," *IEEE Network*, Vol. 13, Sep/Oct 1999, pp. 8-16
- [VE91] D. Verma, H. Zhang, and D. Ferrari, "Delay jitter control for real-time communication in a packet switching network," in *Proceedings of IEEE TriCom 1991*, pp. 35-43
- [WA01] W. Wang and C-C. Shen, "An Adaptive Flow Classification Scheme for Data-Driven Label Switching Networks," *ICC*, June 2001, pp. 2613-2619
- [WR97] J. Wroclawski, "The Use of RSVP with IETF Integrated Services", Request for comments, rfc 2210, (proposed standard), Internet Engineering Task Force, September 1997.
- [WU01] J. Wu, M. Hassan, and S. Jha, "QPD: A packet discard mechanism for multimedia traffic in the internet," *IEICE Trans. Comm.*, pp 538-541, 2001.
- [YA03] Y. R. Yang, Min Sik Kim and Simon S. Lam, "Transient behaviors of TCP-friendly congestion control protocols", *Computer Networks*, Vol 41, Issue 2, 5 February 2003, Pages 193-210
- [YAM02] K. Yamakoshi, K. Nakai, E. Oki and N. Yamanaka, "Dynamic deficit round-robin scheduling scheme for variable-length packets," *Electronics Letters*, Vol. 38, 31 Jan 2002, pp. 148-149
- [YAS04] K. Yasukawa, K-I. Baba and K. Yamaoka, "Dynamic Class Assignment for Stream Flows Considering Characteristics of Non-stream Flow Classes," *IEICE Trans. Comm.*, Vol. E87-B, no. 11, November 2004, pp. 3242-3254
- [YAV99] R. Yavatkar, Don Hoffman, Yoram Bernet, Fred Baker, Michael Speer "Subnet Bandwidth Manager: A protocol for RSVP-based Admission Control over 802-style networks" Internet Draft (work in progress) draft-ietf-issll-is802-sbm-09.txt .1999
- [ZH93] H. Zhang and D. Ferrari, "Rate-Controlled Static-Priority Queueing," in *Proc. of IEEE INFOCOM*, (San Francisco, CA), pp. 227--236, Apr. 1993

- [ZH95] H. Zhang, "Service discipline for guaranteed performance in packet switching networks", proceedings of the IEEE, Vol. 83, Oct 1995, pp. 1374-1396
- [ZHL90] L. Zhang, "VirtualClock: A New Traffic Control Algorithm for Packet Switching Networks," Proc. SIGCOMM 90, Sept. 1990, Philadelphia, PA, pp. 19—29
- [ZHE99] J. Zheng, Victor O.K. Li and X.C. Yuan, "An Adaptive Flow Classification Algorithm for IP Switching," IEEE Global Telecommunications Conference – Globecom'99, pp. 1290-1294